A VIRTUAL TEST BED FOR PEBB-BASED SHIP POWER SYSTEMS

ANNUAL TECHNICAL REPORT

Report # VTB9706001

June 1997

Submitted

by

Roger Dougal, *Project Director*
on behalf of the project team

ONR Grant N00014-96-1-0926

Volume I

University of South Carolina
Dept. of Electrical and Computer Engineering
Columbia, SC 29208

PREFACE

This technical report is primarily a compilation of presentations made at the annual project review meeting held on June 3-4, 1997 at the University of South Carolina. Additionally, this report includes an introduction to the project, and copies of papers that were written under the sponsorship of this grant during the past year.

Any questions regarding the content of this report or specifics of the project can be addressed to

Dr. Roger A. Dougal
VTB Project Director
Dept of Electrical and Computer Engineering
University of South Carolina
Columbia, SC 29208

phone: (803) 777-7890
fax: (803) 777-8045
email: dougal@ece.sc.edu

INDEX

PUBLICATIONS INDEX

Volume III

"A time domain model for flicker analysis", A.P. Sakis Meliopoulos and G. J. Cokkinides, IPST '97, October 1997.

"Wavelet-based transient analysis", A.P. Sakis Meliopoulos and Chien-Hsing Lee, 29th North American Power Symposium, Oct 13-14, 1997.

"An efficient and accurate method of incorporating magnetic saturation in the physical-variable models of synchronous machines" S. D. Pekarek and E.A. Walters,

"A fast and efficient multi-rate technique for detailed simulation of AC/DC power systems", S. D. Pekarek, O. Wasynczuk, H.J. Hegner,

"Digital tracking control for PWM systems with unacceptable zeros", M. Al-Numay and D. Taylor, submitted to IEEE Trans. on Circuits and Systems - special issue on Simulation, Theory, and Design of Switched-analog Networks.

"Adaptive control of DC motor drives with converter nonlinearities" W. Khan and D. Taylor, submitted to Intern. Journal of Control

"Adaptive control of DC motor drives with inverter nonlinearities" W. Khan and D. Taylor, submitted to Intern. Journal of Control

"Modeling and control of digital PWM systems using averaging" M. Al-Numay and D. Taylor, submitted to IEEE Trans. on Control Systems Technology

"Polymer current limiters for low-voltage power distribution systems", M. H. McKinney, C.W. Brice, and R.A. Dougal, IEEE Conf on Industrial and Commercial Power Systems, May 1997, Philadelphia, PA.

"Global Asymptotic stability of indirect field-oriented speed control of induction motors", L. Gokdere, M. A. Simaan, and C. W. Brice, submitted to Automatica

"A passivity-based controller for saturated induction motors" L. Gokdere, M. A. Simaan, and C. W. Brice, submitted to IEEE Trans on Control Sys Tech

"Incorporation of magnetic saturation effects into passivity-based control of induction motors", L. Gokdere, M. A. Simaan, and C. W. Brice, submitted to IEEE Trans. on Industrial Electronics

"A comparison of passivity-based and input-output linearization controllers for induction motors" L. Gokdere, M. A. Simaan, and C. W. Brice, accepted for presentation at IEEE Emerging Technologies and Factory Automation Conf., Sept. 9-12, 1997, Los Angeles, CA.

"Speed estimators for indirect field-oriented control of induction motors" L. Gokdere, M.A. Simaan, and C.W. Brice, accepted for IEEE Emerging Tech and Factory Automation Conf., Sept. 9-12, 1997, Los Angeles, CA.

"A passivity-based controller for high-performance motion control of induction motors", L. Gokdere, M. A. Simaan, and C. W. Brice, accepted for presentation at IEEE Power Electronics Specialists Conf, June 22-2- St. Louis, MO.

INTRODUCTION

The Power Electronic Building Block (PEBB) will enable the Navy to meet DOD and Navy goals of reduced manning, reduced cost, increased effectiveness and enhanced survivability by allowing radically new architectures for shipboard power systems. Since these new architectures cannot be based on historic design precedents and time-and-field-tested design rules, extensive prototyping and testing are necessary. These prototypes are used to validate the designs and to define the operational envelope, in both intact and damaged conditions. The use of virtual prototypes, rather than hardware prototypes, allows the US Navy to maintain its technological superiority by exploiting its dominant position with respect to information technologies.

The Virtual Test Bed (VTB) provides a unique capability for virtual prototyping of PEBB devices and of PEBB-based electric power systems by integrating into a single simulation environment models that have been produced in a variety of simulation languages by a diversified and multi-technical design team.

Those who develop new pieces of the shipboard power system often (and for good reasons) use different software tools for different modeling tasks. Since a system is composed of many such entities, and the system must be tested as a whole, one encounters the need to compute the performance of a system described by a heterogeneous collection of models. The primary objective of the VTB project is to create the virtual prototyping environment that accepts this heterogeneous collection of models and integrates them into a single simulation so that a design engineer can evaluate and understand the dynamic performance of the entire system. The VTB

- allows closer collaboration amongst experts in different fields
- allows each expert to use the best design tools and best design practices within their own area of expertise
- allows integration of more aspects of the design, including physical configuration, electrical configuration, thermal configuration, etc., into a single virtual prototype
- eliminates the need for manual translations of models while exploring system response
- allows more rapid exploration of a larger design space to yield more optimal designs
- provides more advanced visualizations of system performance to help build an intuitive understanding of the influence of controllable parameters

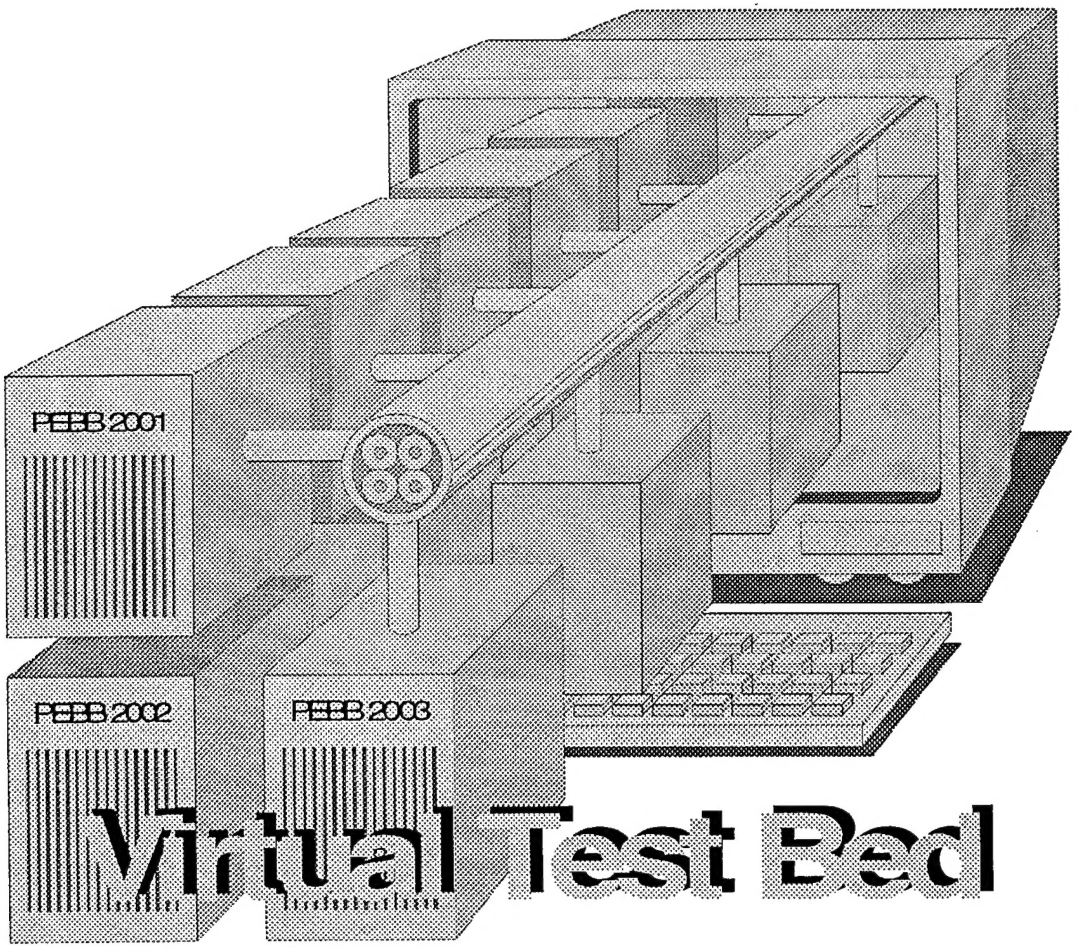
The approach chosen for creating the VTB's mixed-language virtual prototyping environment involves *translation* of the source models into the VTB internal language. This provides a flexible, powerful, robust, and extensible means for integrating models into a unified simulation environment. In addition to developing the model translation technologies, the VTB project also advances other technologies associated with virtual prototyping, including user interfaces, model libraries, execution environments, and visualization tools. These technical advances allow the VTB to fully support development and application of the PEBB.

A secondary objective of the VTB project is to develop a base library of models that can be used in the Virtual Test Bed. These models should execute rapidly, be constructed in such a way that they can be connected to other models, and yet can be written in a variety of modeling/simulation languages. Particularly important to the study of PEBB-based power systems is an accurate model of a PEBB.

The third and final objective of this project is to support development and applications of PEBBs by using existing modeling tools and existing components of the VTB for virtual prototyping of the PEBB before the VTB is fully developed.

The presentation materials that follow describe a majority of the technical work that was performed under the auspices of this grant during the past year. Time constraints during the meeting prevented the presentation of all aspects of all work that was performed, but this report provides a fair cross section and a useful summary.

Somewhat arbitrarily, the report is organized according to the presentation scheme that was used at the annual review meeting. That is, items are arranged according to the site at which work was done, rather than by topic.



Virtual Test Bed

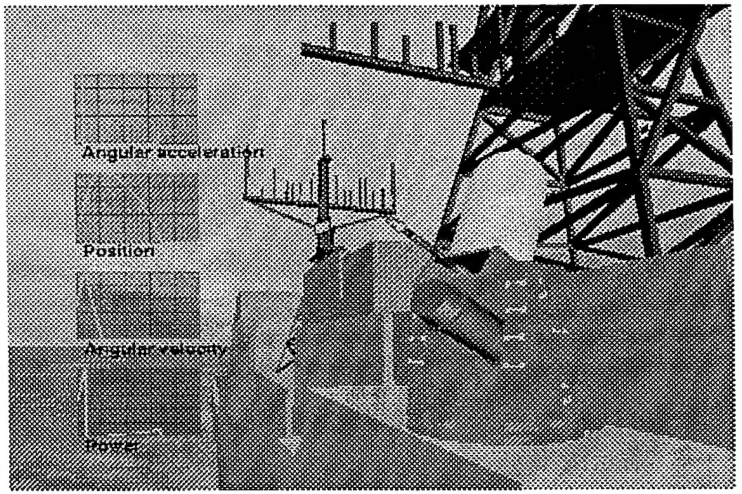
UNIVERSITY OF SOUTH CAROLINA

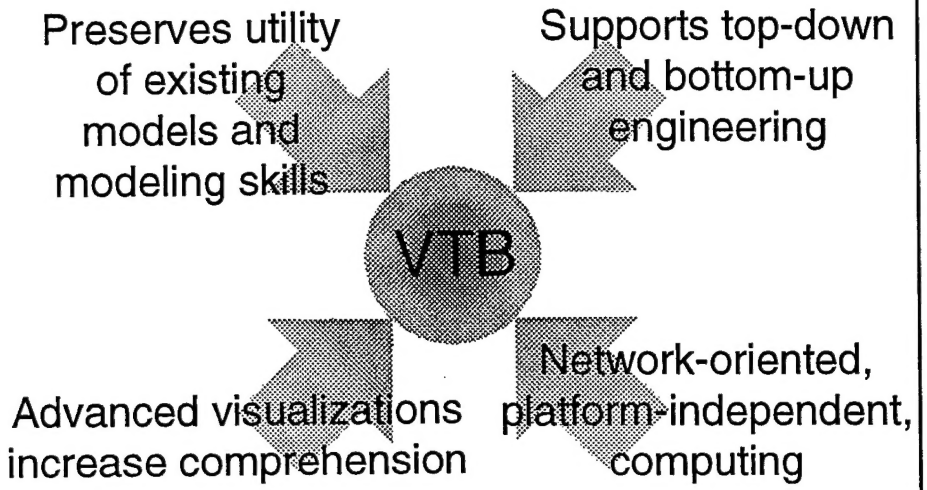
UNIVERSITY OF SOUTH CAROLINA

Virtual Test Bed

Software environment for virtual prototyping of electric power systems

Focus on developing PEBB and on evaluating ship applications of PEBB





- Supports goals of DOD initiatives
  - Simulation Based Design
  - Simulation Based Acquisition
- Supports focussed programs
  - Power Electronic Building Block
  - Autonomic ship
  - Integrated Power System for SC-21

UNIVERSITY OF SOUTH CAROLINA

Virtual Test Bed

Supports rapid iteration of cross-disciplinary designs

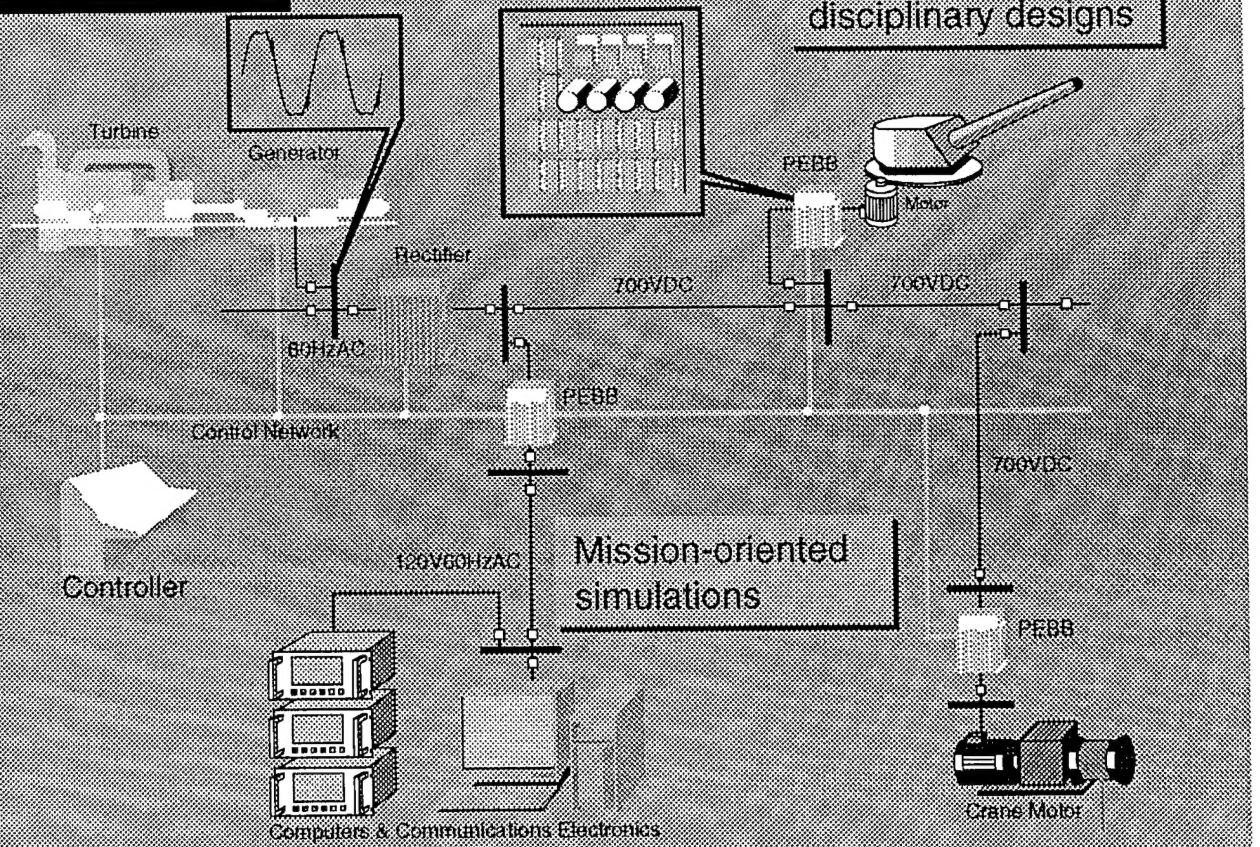
Integrates models across disciplines

PEBB-based power systems

Supports models at multiple levels of detail

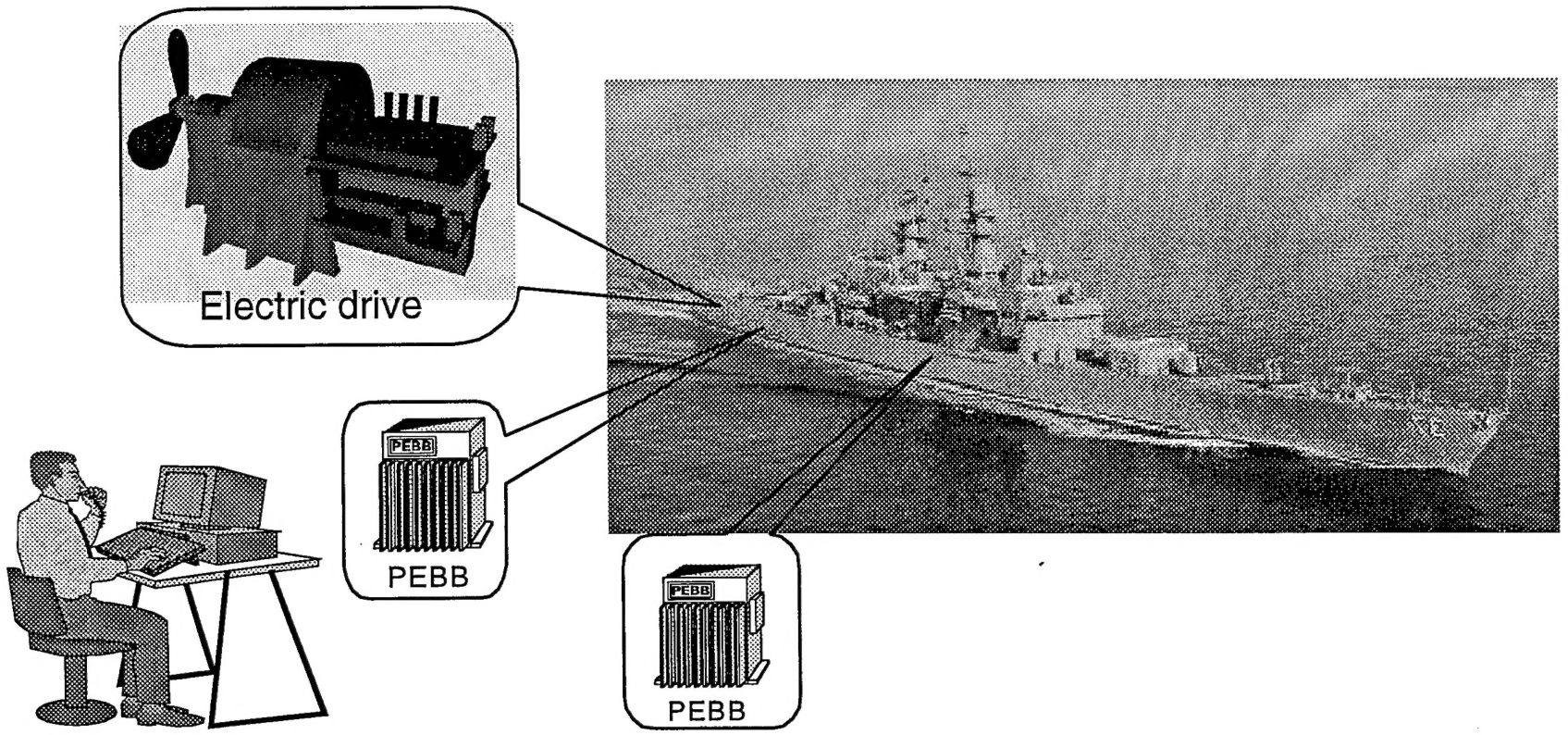
ly cal

Preserves the value of language dependent modeling skills



VTB models can assume multiple levels of detail

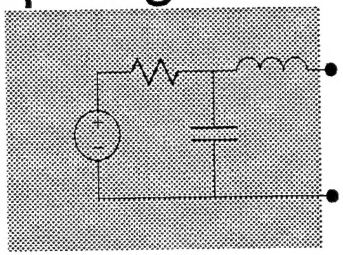
TOP-DOWN analysis supports conceptual design and rapid iteration

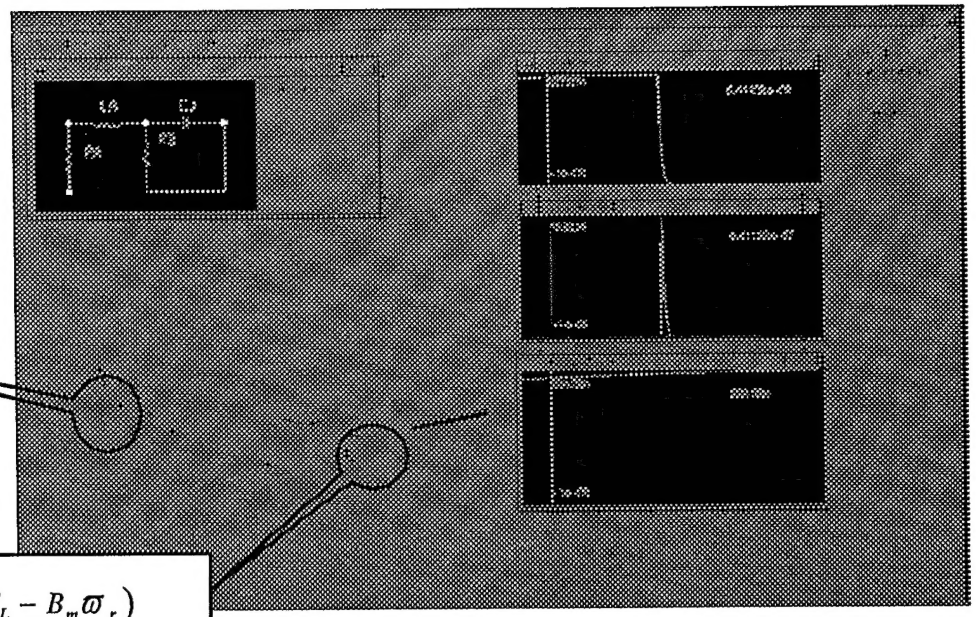


VTB models can assume multiple levels of detail

BOTTOM-UP analysis supports detailed interdisciplinary engineering tasks

Topological model





Procedural model

$$\frac{d\omega_r}{dt} = \frac{1}{J}(T_e - T_L - B_m\omega_r)$$

$$T_e = L_{AF} i_f i_a$$

$$\frac{di_f}{dt} = \frac{v_f - r_f i_f}{L_{FF}}$$

UNIVERSITY OF
SOUTH CAROLINA

Advantages of VTB are similar to those of PEBB

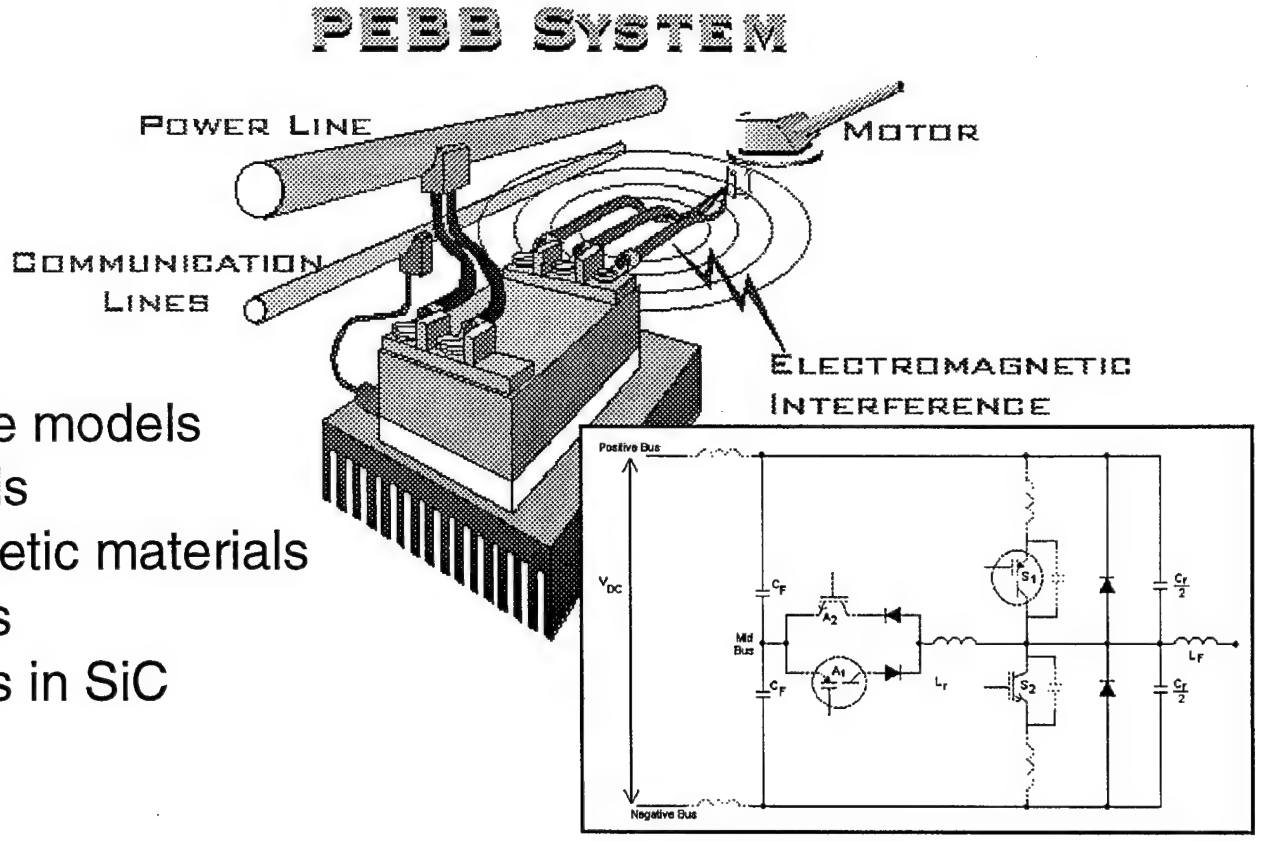
- Removes details of implementation from system modeler
- Opens up modeling environment
  - Models have standard interfaces, so are reusable
  - Appropriate expert can be used for each device model
  - Eliminates technology learning curve for cross-disciplinary work, so more work gets done faster
  - Large jobs can be parceled out to many sites

But further, VTB will eliminate layers of hardware testing and evaluation to yield

- More rapid technology insertion
- More capable equipment in the field
- Lower life-cycle cost

VTB project is more than "just" software development

Building models



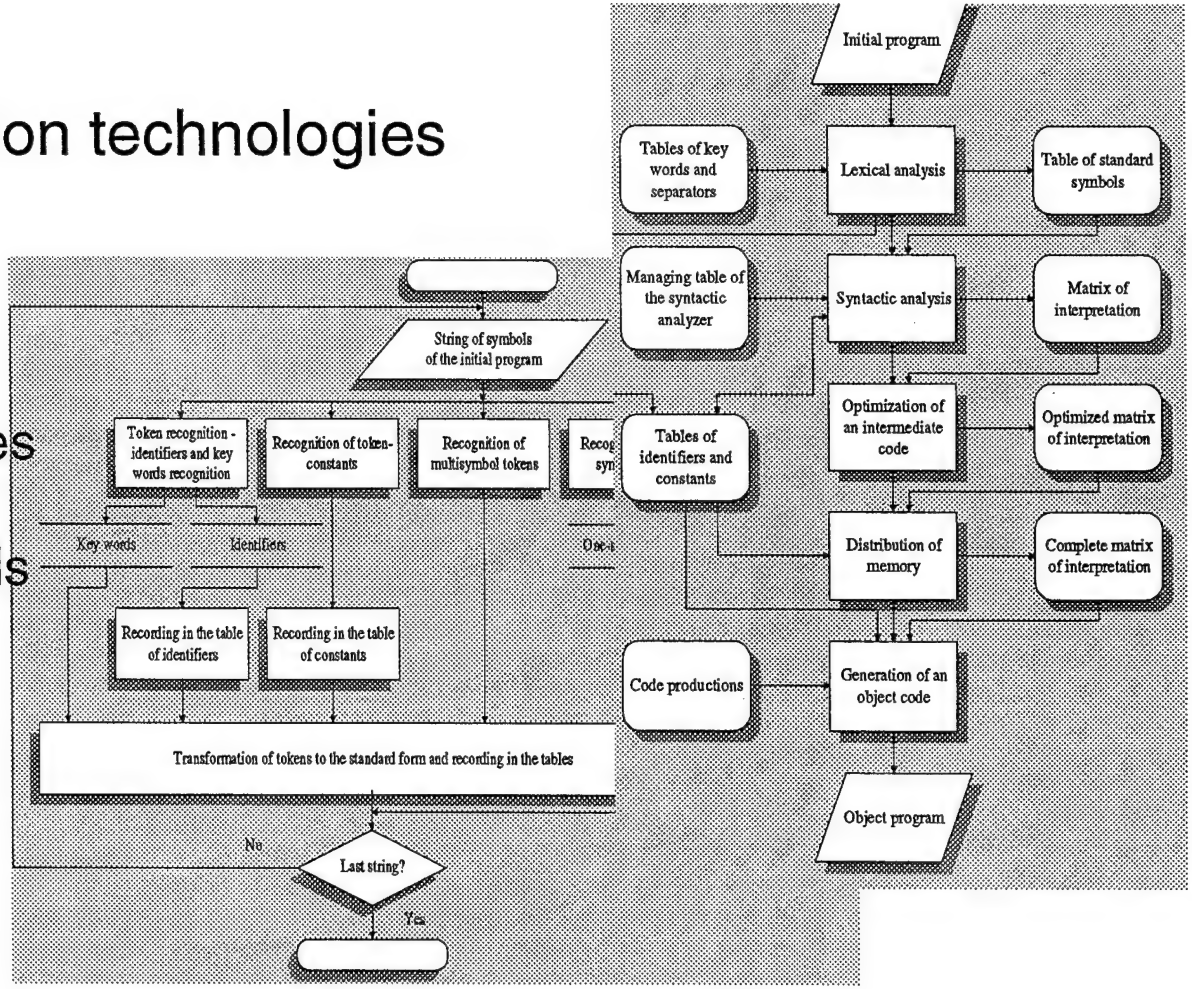
- PEBB power module
- Fast-executing machine models
- Hi Freq Machine models
- Power lines near magnetic materials
- Polymer current limiters
- Potential PEBB devices in SiC

UNIVERSITY OF SOUTH CAROLINA

VTB project is more than "just" software development

Advancing simulation technologies

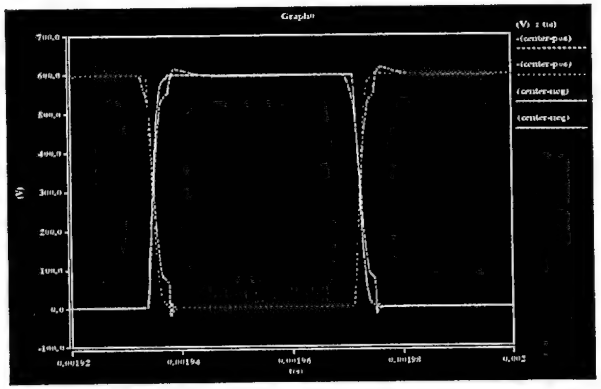
Computational techniques
Partitioning of models
Interconnection of models

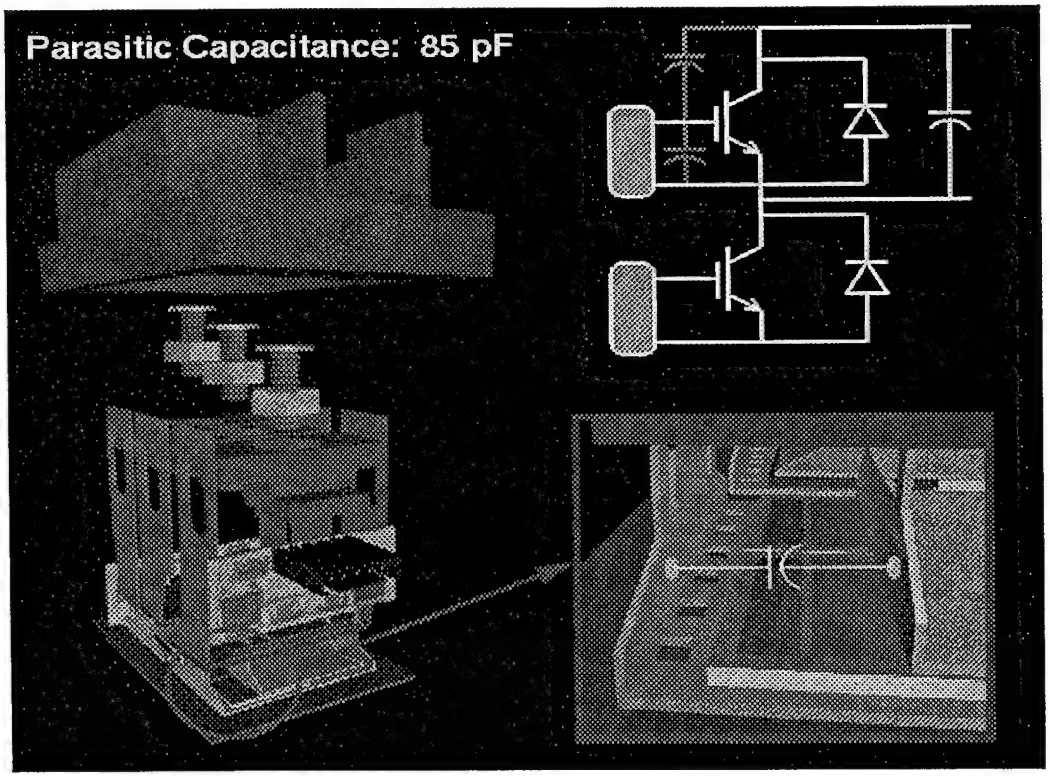


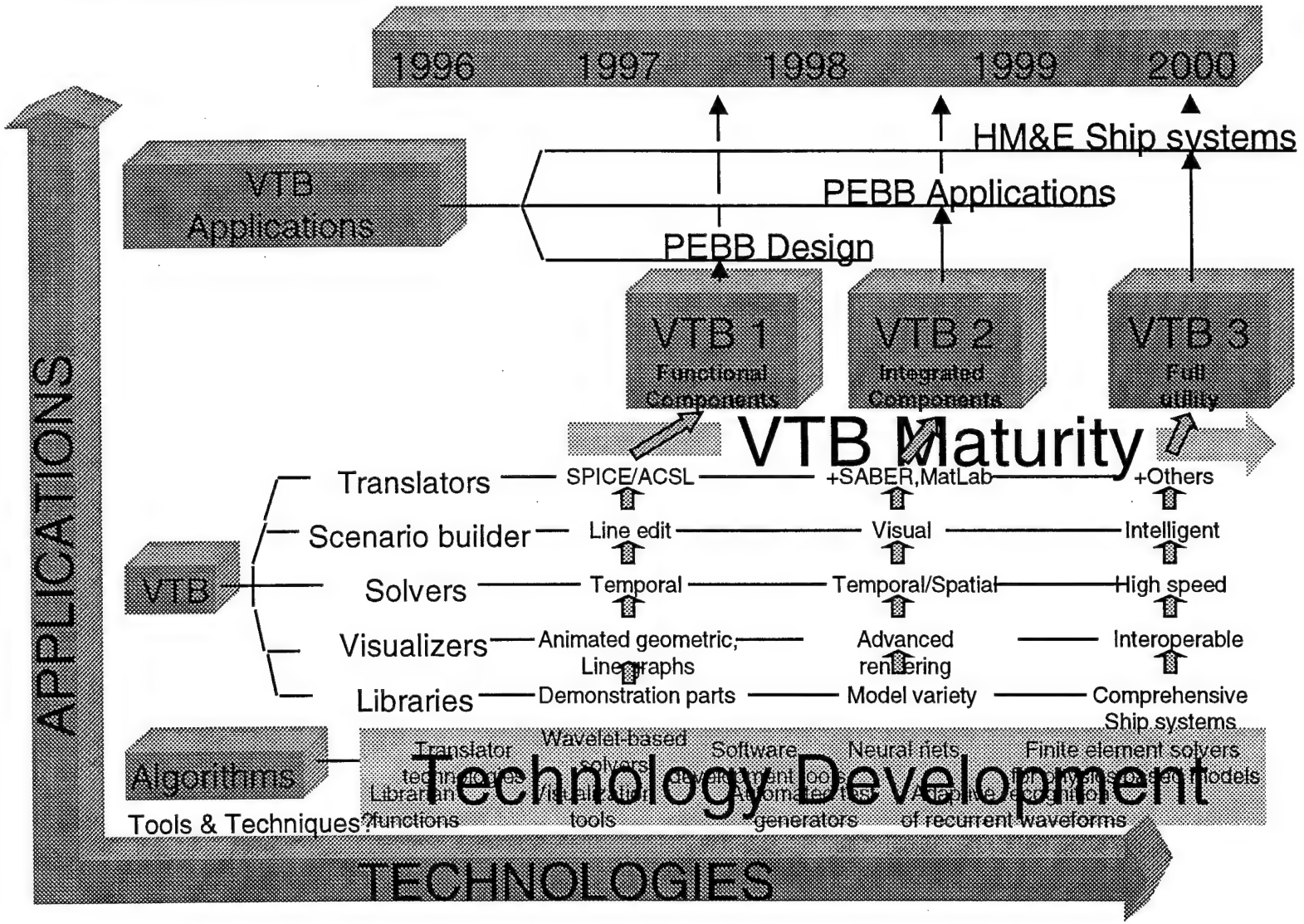
VTB project is more than "just" software development

Application of VTB capabilities to PEBB development

Package characterization
Performance predictions
Design iteration
Component visualization



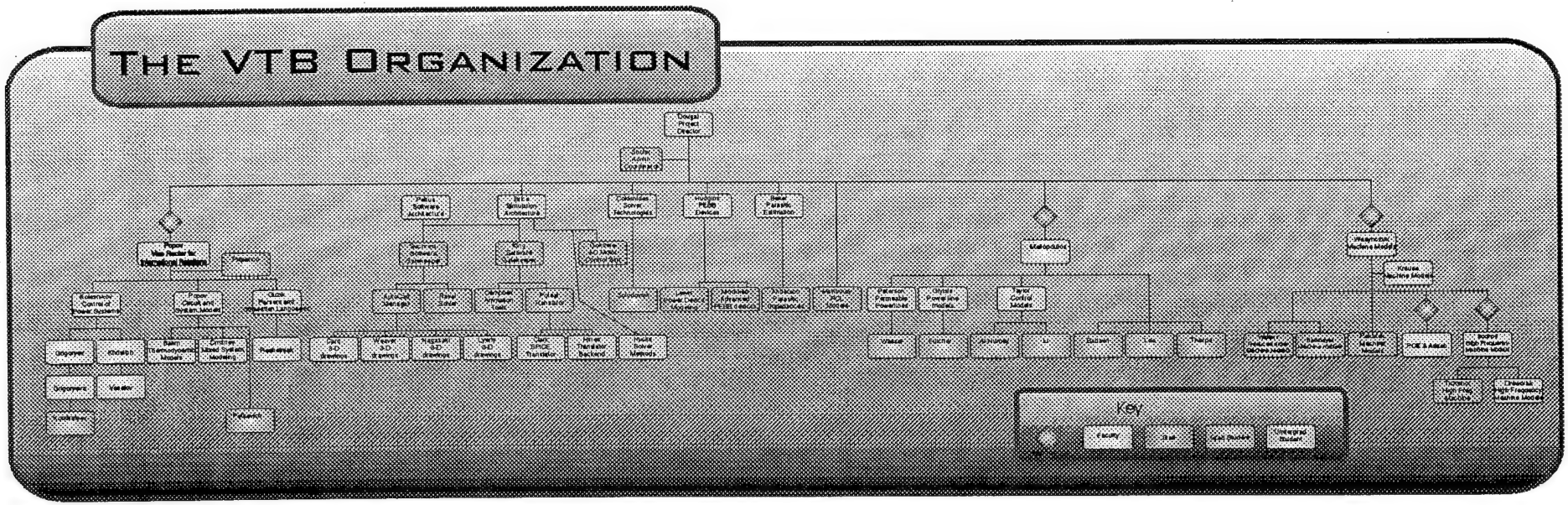


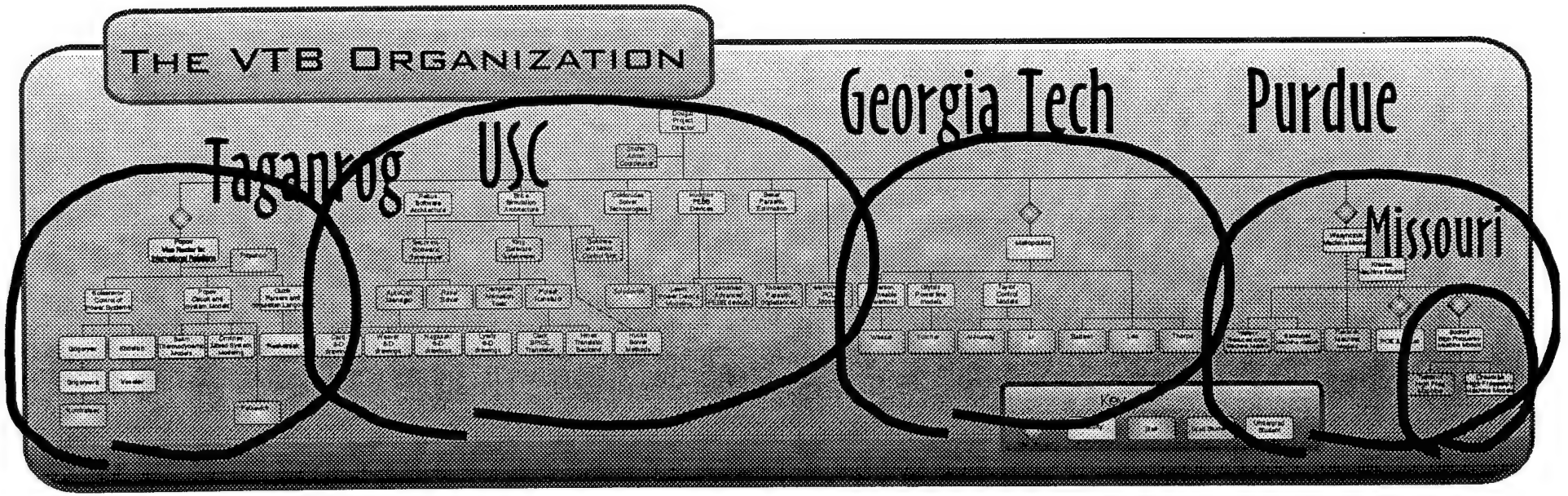

1996
1997
1998
1999
2000
HM&E Ship systems
VTB Applications
PEBB Applications
PEBB Design
VTB 1
Functional Components
VTB 2
Integrated Components
VTB 3
Full utility
VTB Maturity
APPLICATIONS
VTB
Translators
SPICE/ACSL
+SABER,MatLab
+Others
Scenario builder
Line edit
Visual
Intelligent
Solvers
Temporal
Temporal/Spatial
High speed
Visualizers
Animated geometric, Line graphs
Advanced rendering
Interoperable
Libraries
Demonstration parts
Model variety
Comprehensive Ship systems
Algorithms
Translator techniques
Wavelet-based solvers
Software
Neural nets
Finite element solvers
Tools & Techniques
Functions
tools
generators
of recurrent waveforms
Technology Development
TECHNOLOGIES

OFFICE OF NAVAL RESEARCH

UNIVERSITY OF SOUTH CAROLINA

THE VTB ORGANIZATION




THE VTB ORGANIZATION
Taganrog
USC
Georgia Tech
Purdue
Missouri

- Distinguishing characteristics of VTB
  - Model interconnectivity
  - Mission-oriented approach to simulation
  - Geometry visualization tied to performance
- Watch for these during the ensuing presentations

UNIVERSITY OF
SOUTH CAROLINA

TUESDAY JUNE 3

8:00	8:20	Registration, coffee, donuts
8:20	8:25	Introduction
8:25	8:35	Welcome
8:35	8:55	VTB Project Overview
8:55	10:35	VTB IMPLEMENTATION
10:35	10:45	BREAK
10:45	12:00	VTB IMPLEMENTATION
12:00	1:00	LUNCH
1:00	2:20	Georgia Tech reports
2:20	3:50	DEMONSTRATIONS
3:50	4:50	Taganrog State University reports
4:50	5:00	Summary

UNIVERSITY OF
SOUTH CAROLINA

WEDNESDAY JUNE 4

7:50	8:10	Coffee & Donuts
8:10	9:10	Purdue/Missouri reports
		PEBB APPLICATIONS of VTB Technologies
9:10	9:25	SiC Power devices
9:25	9:40	PEBB Parameter estimation
9:40	9:55	PEBB 2 Package
9:55	10:10	ARCP model
10:10	10:30	Georgia Tech
10:30	10:40	BREAK
10:40	11:50	FOCUS GROUPS
		Computational performance
		Model standards
		Visualization tools
		Evolution of simulation technology
11:50	12:50	LUNCH
		Focus group reports
12:50	1:05	Computational performance
1:05	1:20	Model standards
1:20	1:35	Visualization tools
1:35	1:50	Evolution of simulation technology
1:50	2:15	Plans for Year 2
2:15	2:35	Navy feedback
2:35	2:55	Action plans
2:55	3:20	Wrap Up

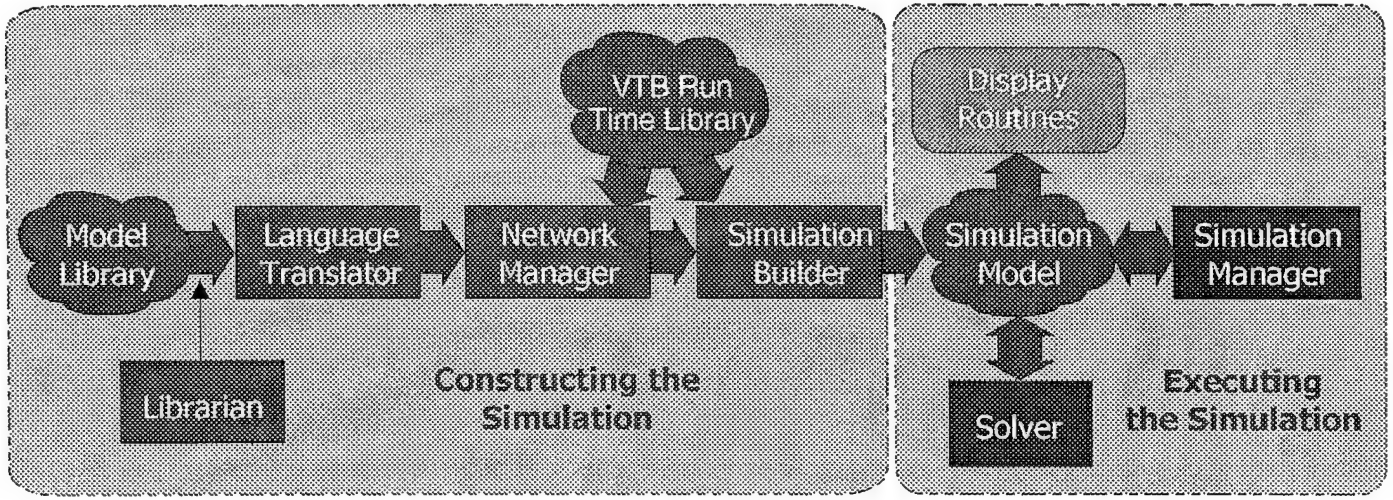
VTB PRESENTATION

- VTB Architecture
- Technical Presentations (VTB Implementation)
- Assessment and Future Directions

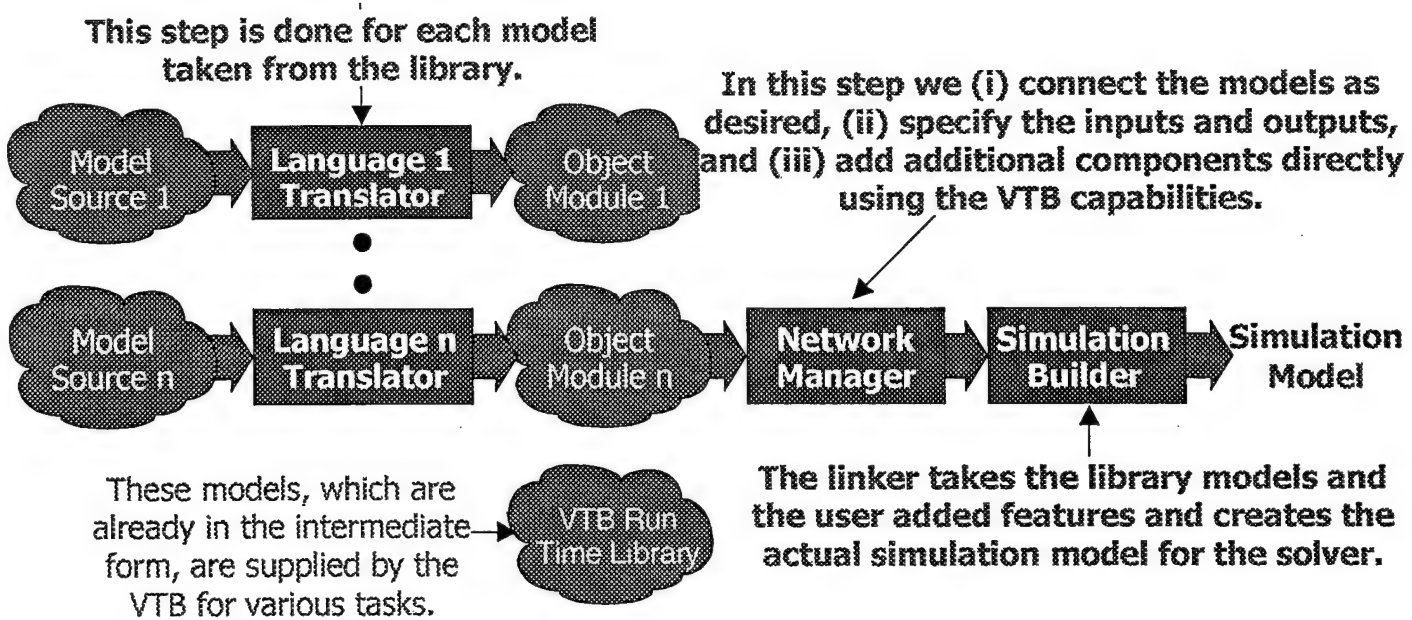
VTB ARCHITECTURE

- Overall structure and organization
- Methods — algorithms & data structures
- User interface — look & feel
- Implementation issues
- Performance issues

INITIAL ARCHITECTURE



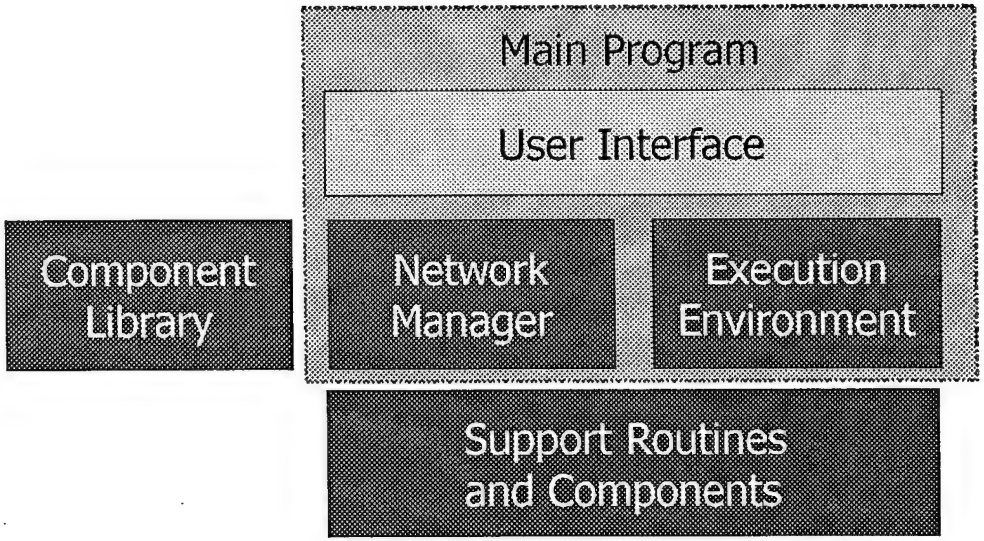
TRANSLATORS



VTB EVOLUTION

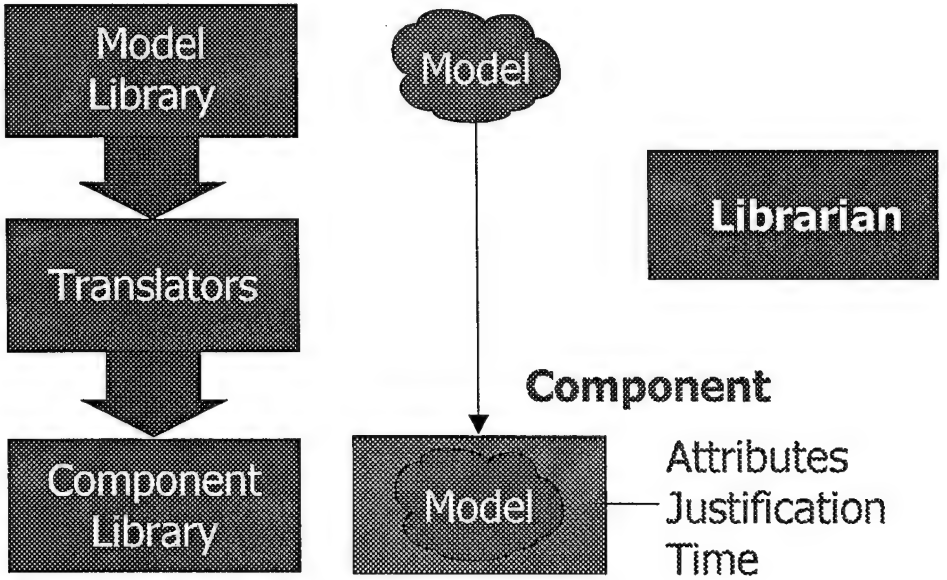
- The current architecture has been refined, but not substantially changed.
- The translators are being implemented in a serial fashion.
- Definition of the solver interface was the number one issue.

VTB COMPONENTS



The component library is designed to be an independent too.

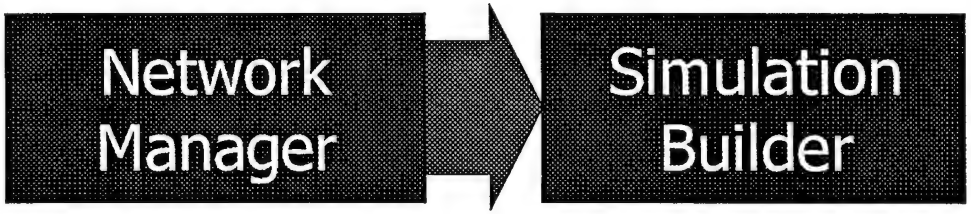
COMPONENT LIBRARY



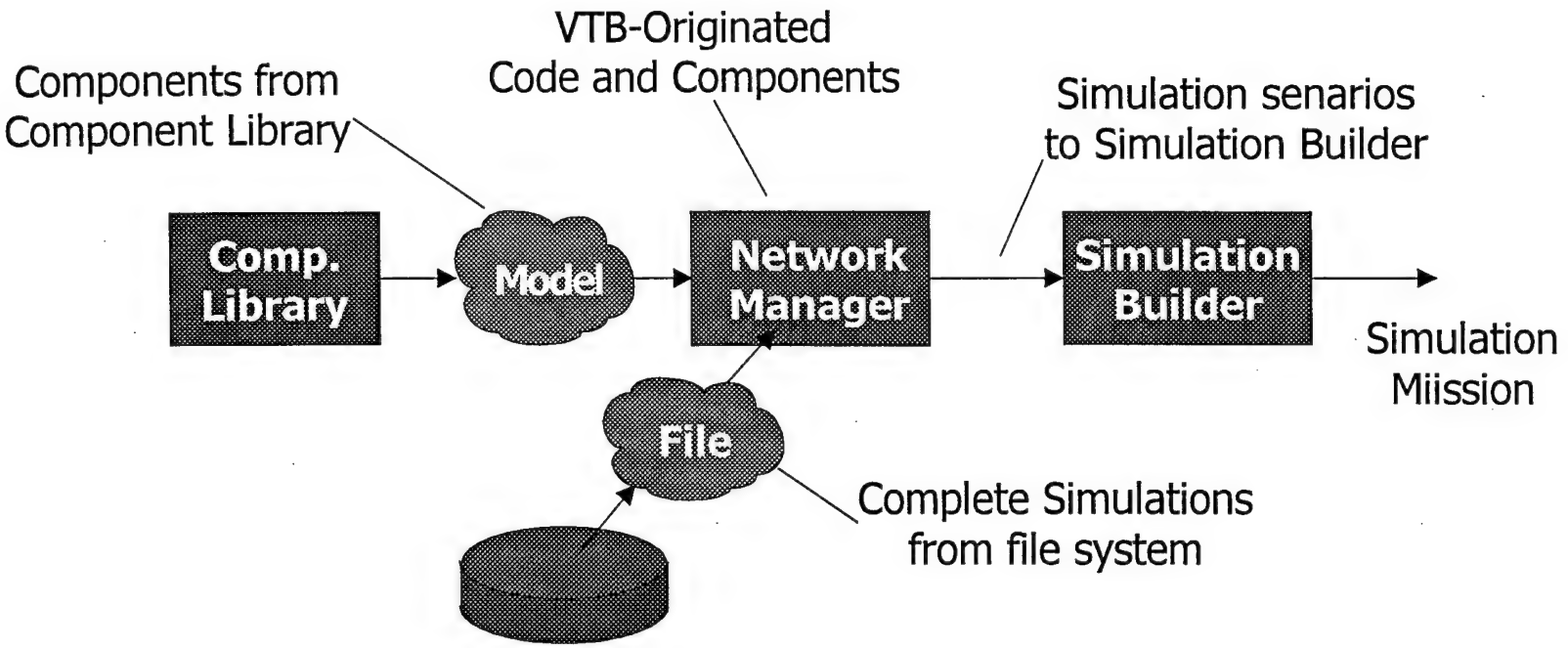
NETWORK MANAGER

Tool used to create and edit the simulation scenario

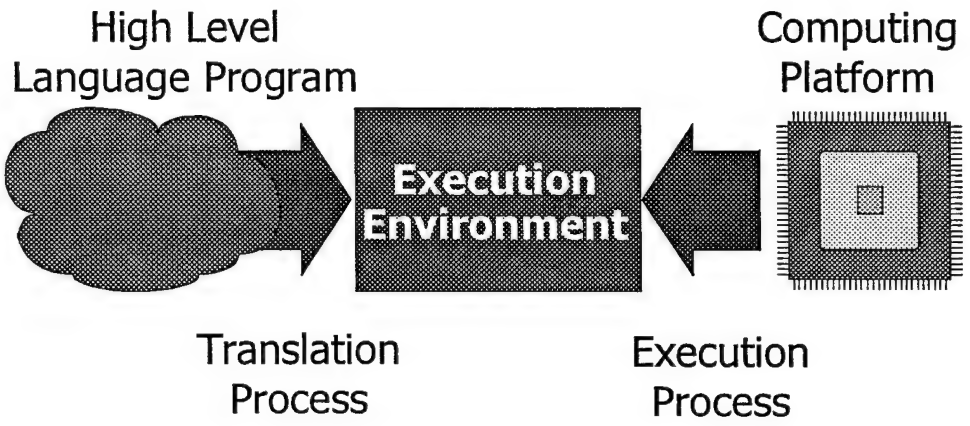
System tool used to link multiple components to create an executable file



NETWORK MANAGER 2

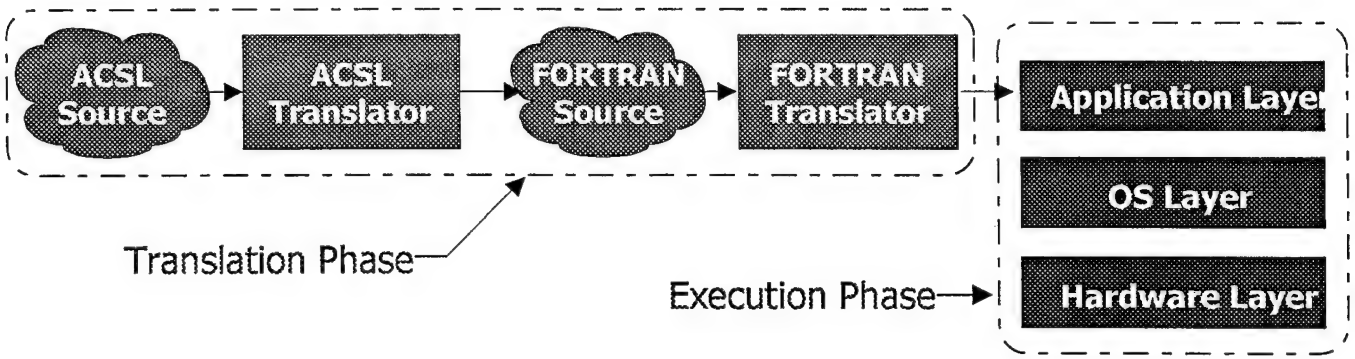


EXECUTION ENVIRONMENT

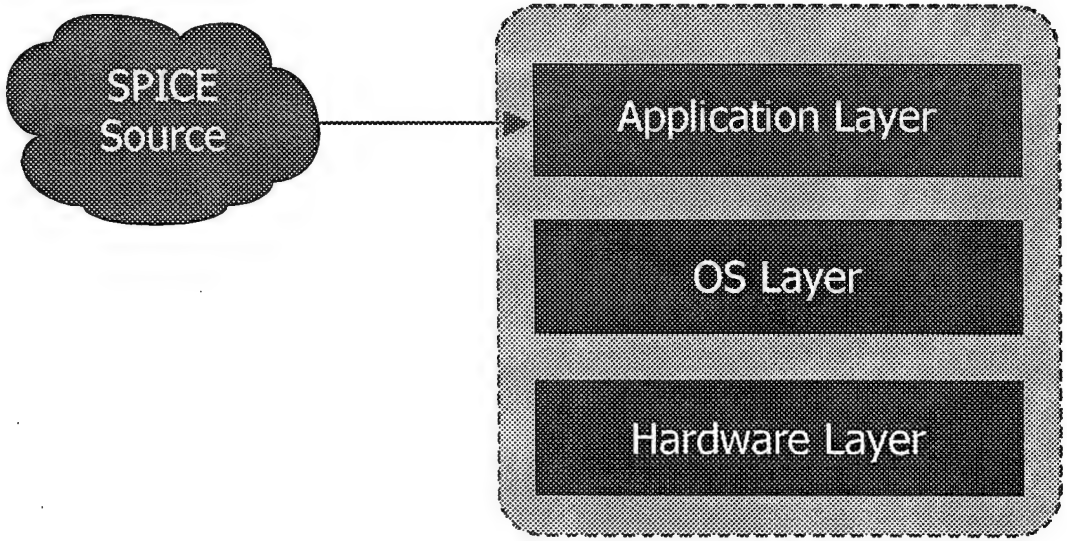


The execution environment bridges the gap between the platform (hardware) and application (software).

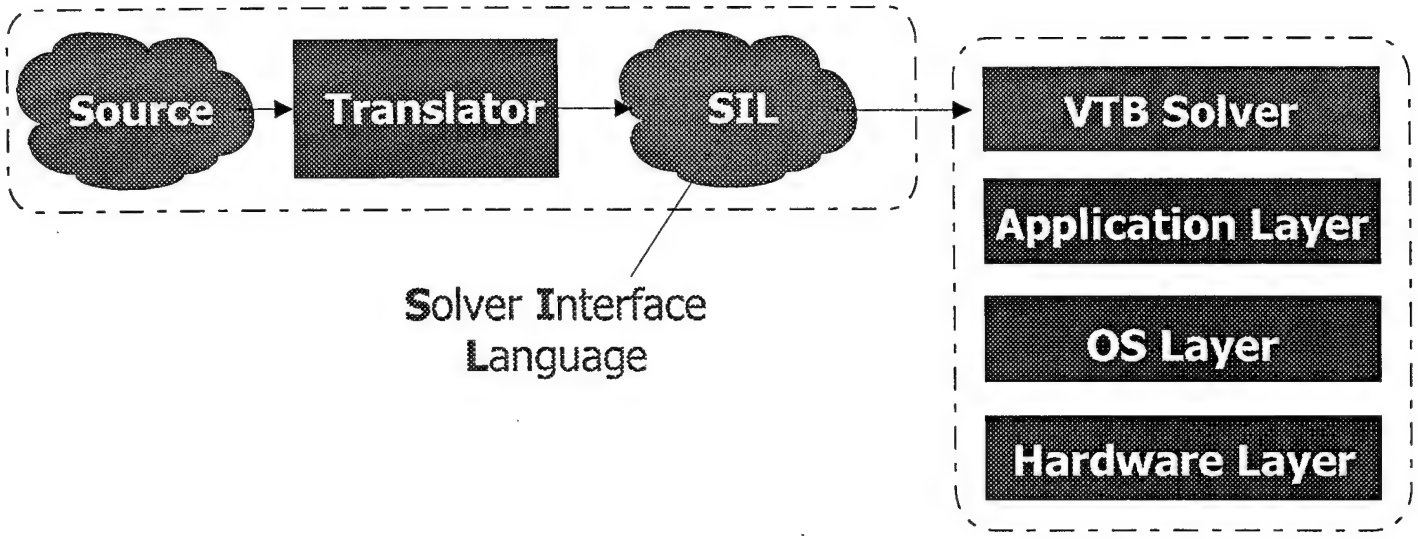
ACSL ENVIRONMENT



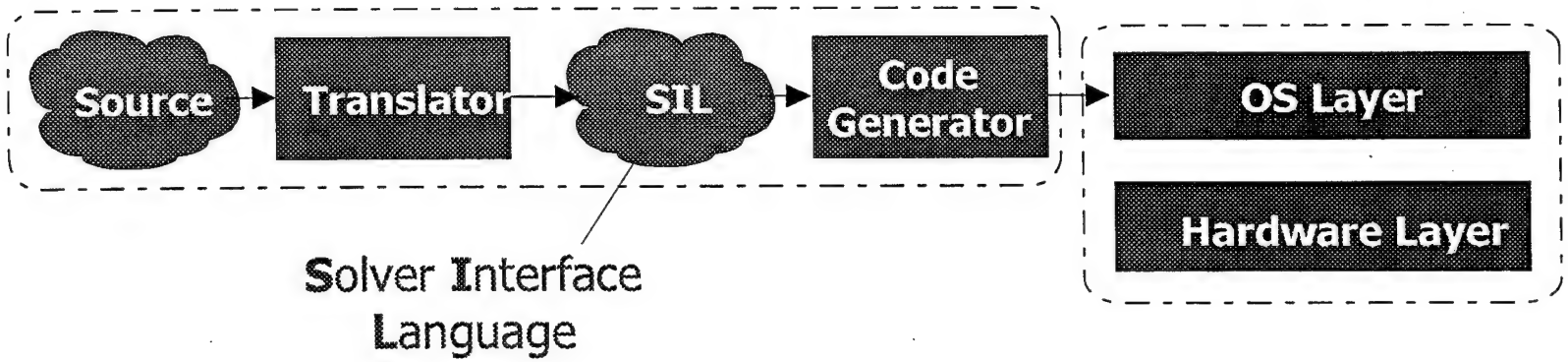
SPICE ENVIRONMENT



VTB ENVIRONMENT - 1



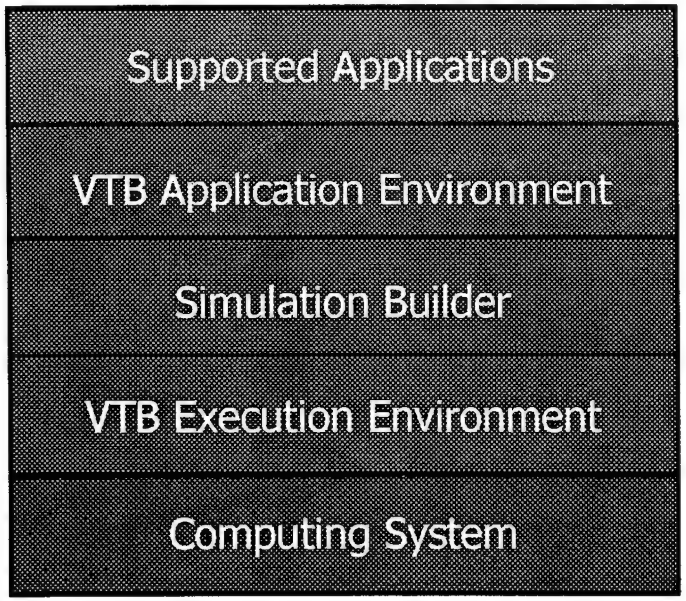
VTB ENVIRONMENT - 2



SOLVER INTERFACE LANGUAGE (SIL)

- Common language for all supported tools.
- Designed for 1) efficient execution by solver and 2) use as intermediate language.
- Level at which all components communicate.
- Allows custom features for each tool.

VTB HIERARCHY



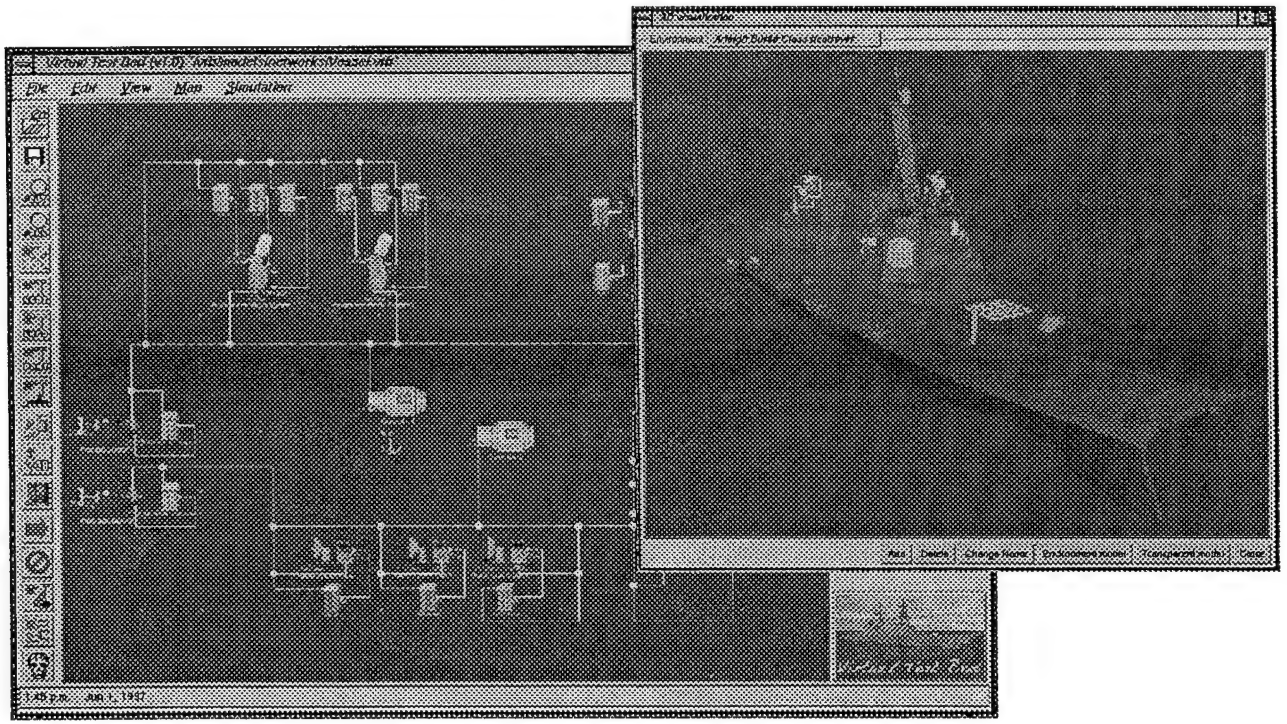
Solver algorithms, circuit issues, other application-related issues, includes one translator per supported language.

Inter-module communication, time input/output, visualization

LOOK AND FEEL

The primary VTB interface is visual. It supports real-time, 3-D visualization for both programming and display of results. However, the "look and feel" is independent of the implementation and may be chosen to meet application needs.

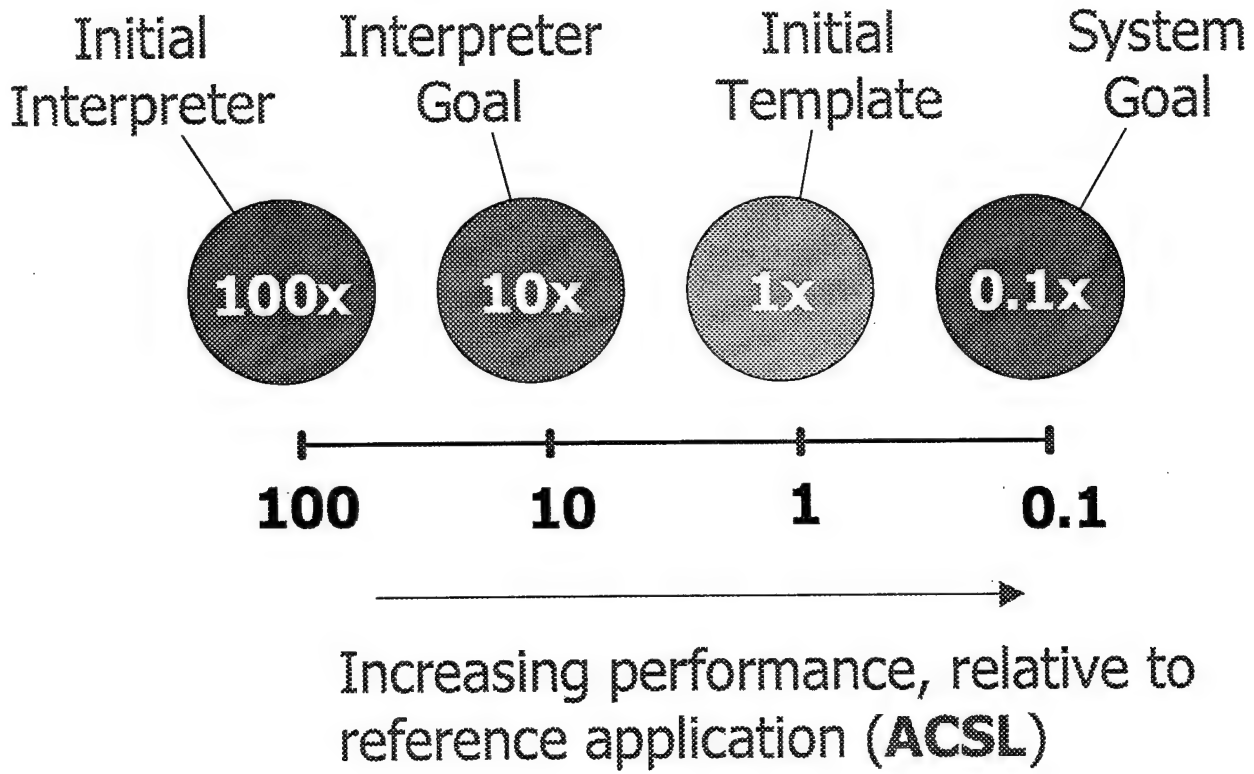
LOOK AND FEEL - 2



REAL-TIME

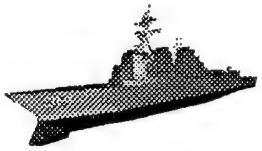
VTB supports real-time operation at both the system and component level. However, each VTB component has its own rules for handling time. These rules are determined when the component is placed in the library, or at time for entry for directly entered objects. Objects may also inherit time from other objects.

PERFORMANCE



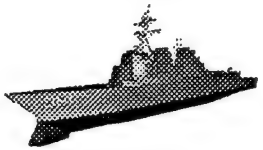
VIRTUAL TEST BED

Component Librarian



OUTLINE

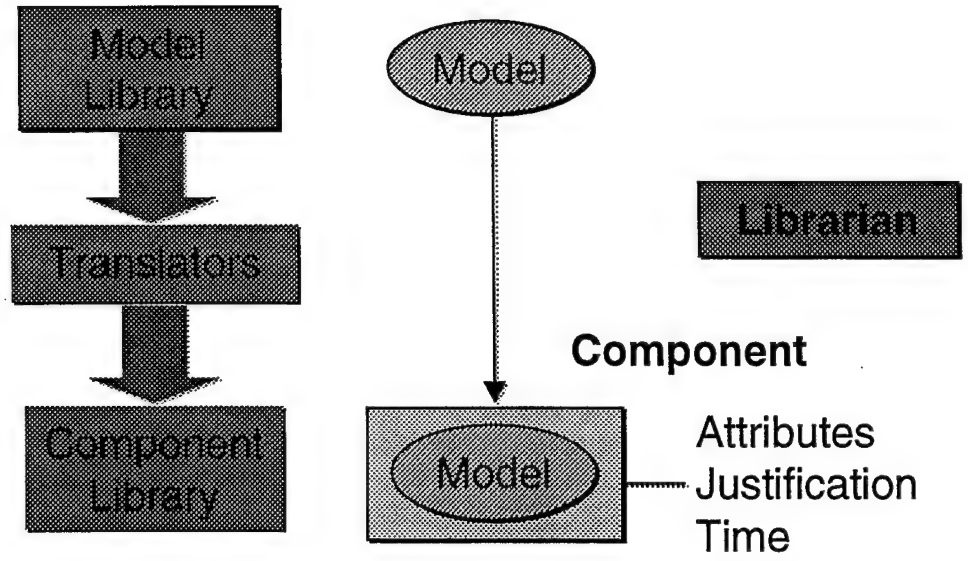
- Role of the component librarian
- "Component" definition
- Librarian user interface
- VTB system interaction

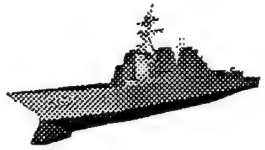


PURPOSE

- Provide a graphical user interface for managing components (add, delete, edit, and retrieve)
- Provide a software API for importing components
- Provide convenience routines to the network manager

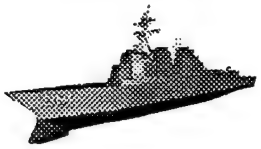
SOFTWARE ARCHITECTURE





COMPONENT ELEMENTS

- Port specifications
- Diagrams
- Visual attributes
- User comments
- Representative models (ACSL, SPICE, SIL, etc.)



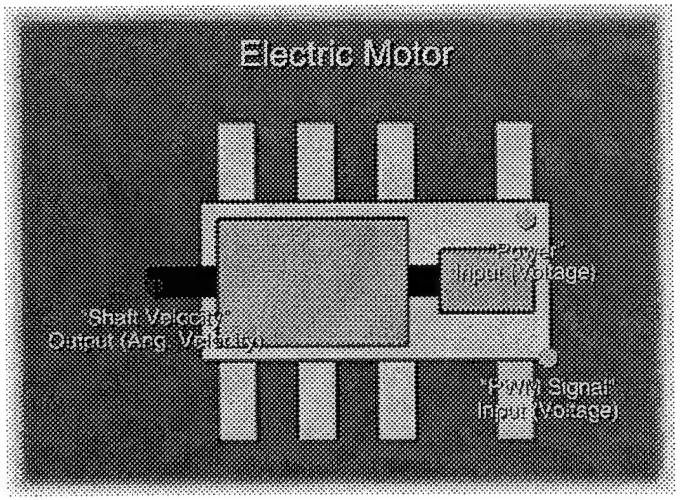
PORTS

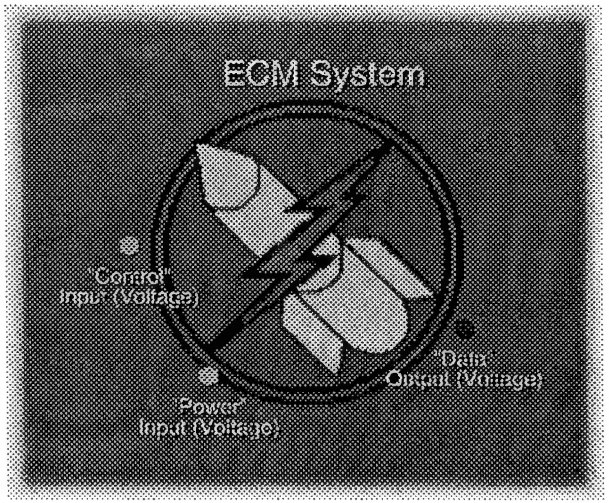
- Ports define the external linkage of a component and consist of
  - I/O Type:

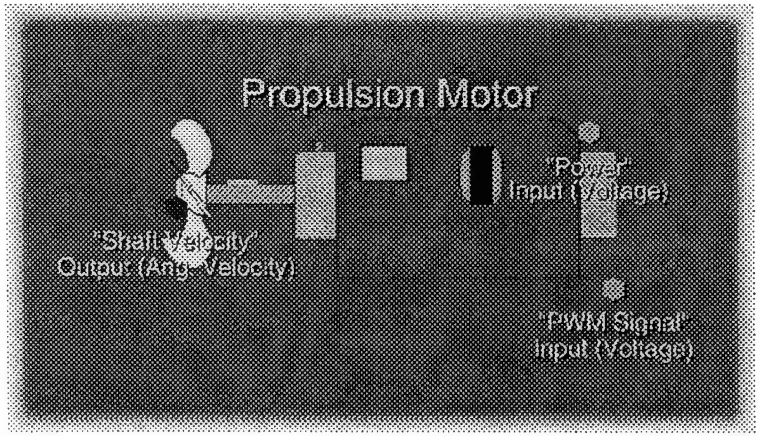
    Input Output Input-Output
  - Physical Type:

Current	Voltage	Velocity
Angular Velocity	Force	Acceleration
Torque	Angular-Acceleration	

ICONS

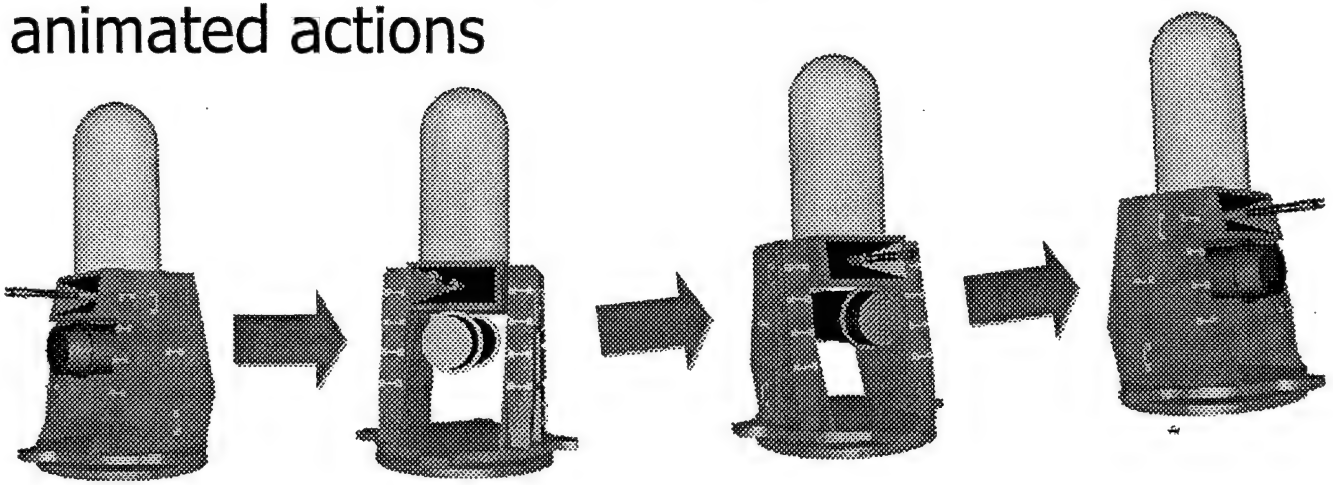

Electric Motor
"Power"
Input (Voltage)
Shaft Velocity"
Output (Ang. Velocity)
"PWM Signal"
Input (Voltage)


ECM System
"Control"
Input (Voltage)
"Data"
Output (Voltage)
"Power"
Input (Voltage)


Propulsion Motor
"Power"
Input (Voltage)
"Shaft Velocity"
Output (Ang. Velocity)
"PWM Signal"
Input (Voltage)

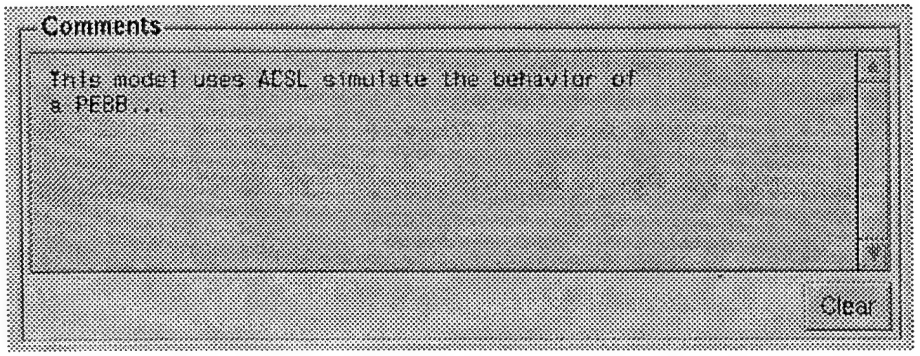
VISUAL ATTRIBUTES

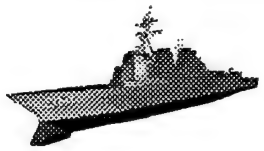
- Visual attributes enable the component to be used in a visual simulation.
  - Three-dimensional model (DXF, Inventor, Alive creature, etc.)
  - Expressions relating simulation variables to animated actions



USER COMMENTS

- Provides a field for user description of component
- Allows for a thorough search of the component database





MODELS

- Each component will have one or more models of varying resolution
- Each of these models will contain
  - Name
  - Resolution metric
  - Model filename (e.g. ACSL source filename)
  - Port mappings
  - User comments (specific to this model)
  - Miscellaneous parameters

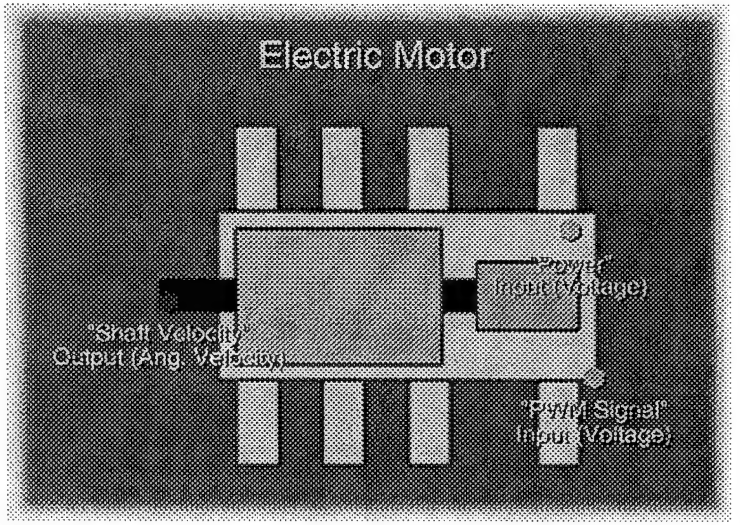
MODEL BACKPLANE

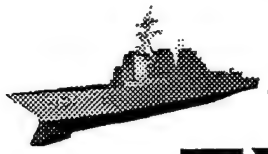
- Backplane provides linker information for the model translators
- Librarian embeds backplane information given through model editor interface

```
!       ACSL Test File
!
!       BackplaneBegin
!               BackplaneInput BackplaneAngularAcceleration -10.0..20.0 x
!               BackplaneInput BackplaneAngularVelocity -1000.0..2000.0 y
!               BackplaneOutput BackplaneVoltage -777.7..888.8 z
!       BackplaneEnd
!

PROGRAM MyACSLProgram
        REAL    x, y, z

        ! simple nonsense equation
        z = x * y

END
```

PORT MAPPINGS

- Each variable exported in the backplane section of the model is mapped to a port defined in the component description
- All topological connections within the network manager will be through these ports





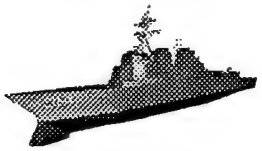
EXAMPLE COMPONENT FILE

```
VTBComponentFile

Component "Motor IM2250"
        {
        Ports
                {
                Port "Power"
                        {
                        GeometricOffset = (-0.33, 0.13);
                        IOType = INPUT;
                        PhysicalType = VOLTAGE;
                        }
                Port "PWM Signal"
                        {
                        GeometricOffset = (-0.33, -0.14);
                        IOType = INPUT;
                        PhysicalType = VOLTAGE;
                        }
                Port "Shaft Speed"
                        {
                        GeometricOffset = (0.32, 0.0);
                        IOType = OUTPUT;
                        PhysicalType = ANGULAR_VELOCITY;
                        }
                }
        Comments
                {
                "This is a model of a 2250 HP
                 induction motor"
                }
```

```
        Icon
                {
                Filename = "generic_motor.iv";
                Scale = 20;
                }
        Model "IM2250" (1)
                {
                Model
                        {
                        Filename = "im2250b.csl";
                        }
                Physical
                        {
                        Creature = "generic_motor.cf";
                        Action "shaftrot" Loop "Speed" (0, 2000) (0, 10);
                        }
                PortMappings
                        {
                        "Shaft Speed" = "Speed";
                        "Power" = "Vsc";
                        "PWM Signal" = "Vpwm";
                        }
                Comments
                        {
                        "ACSL file to be used as for a simple
                         induction motor"
                        }
                }
}
```

COMPONENT FILE BNF

```
<component file>      ::= <file identifier> <component>;
<file identifier>     ::= "VTBComponentFile";
<component>           ::= "Component" <string> '{' <port_mappings>
                          <icon> <comments> { <model> } '}';

<port_mappings>       ::= "PortMappings" '{' { <port> } '}';
<port>                ::= "Port" <string> '{' <geometric offset> <io type>
                          <physical type> '}';
<geometricoffset>     ::= "GeometricOffset" <assign_op> <vector 2D> ';';
<io type>             ::= "IOType" <assign_op> <io value> ';';
<io value>            ::= "INPUT" | "OUTPUT" | "INPUT_OUTPUT";
<physical type>       ::= "PhysicalType" <assign_op> <physical value> ';';
<physical value>      ::= "CURRENT" | "VOLTAGE" | "VELOCITY" |
                          "ANGULAR_ACCELERATION" | "FORCE" | "ACCELERATION" |
                          "TORQUE" | "ANGULAR_VELOCITY";

<icon>                ::= "Icon" '{' <filename> <scale> <units> '}';
<scale>               ::= "Scale" <assign op> <float> ';';
<units>               ::= "Units" <assign op> <units value> ';';
<units value>         ::= "FEET" | "INCHES" | "MILLIMETERS" | "CENTIMETERS" |
                          "METERS" | "YARDS";
<comments>            ::= "Comments" '{' <special string> '}';
```



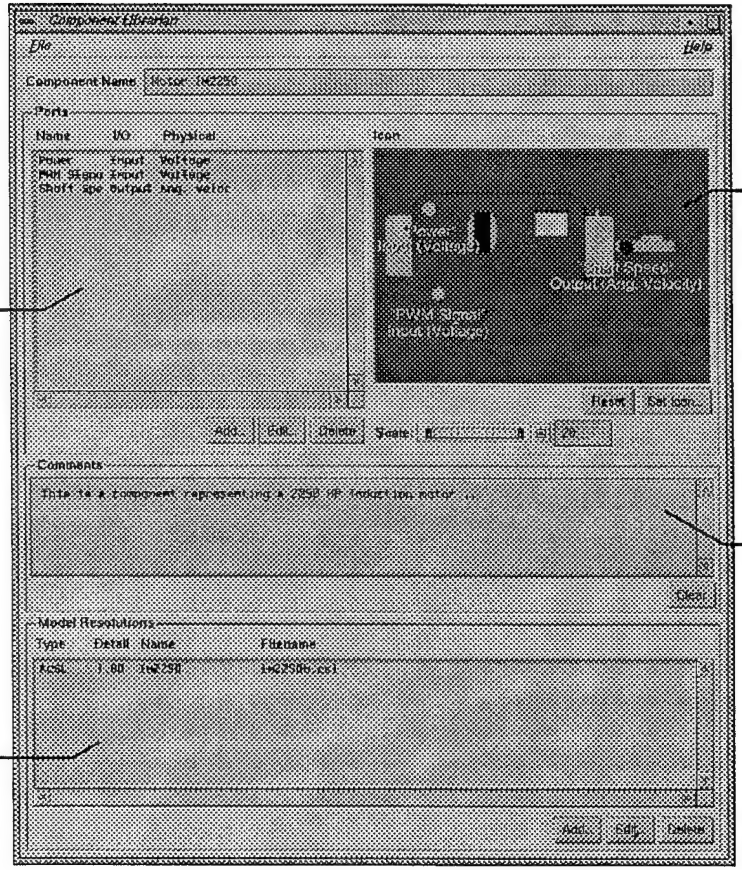
BNF, CONTINUED

```
<model>          ::= "Model" <string> '(' <model metric> ')' '{'
                     <model_file> <port maps> <physical>
                     <comments> '}';
<model metric>   ::= <int>;
<model_file>     ::= "Model" '{' <filename> '}';
<port map>       ::= "PortMappings" '{' { <port entry> } '}'
<port entry>     ::= <string> <assign op> <string> ';';
<parameters>     ::= "Paramters" '{' '}';
<comments>       ::= "Comments" '{' <special string> '}';
<physical>       ::= "Physical" '{' <creature> { <action> } '}'
<creature>       ::= "Creature" <assign op> <string> ';';
<action>         ::= "Action" <string> ( "Loop" | "Bounce" | "Indexed" )
                     <string> <vector 2D> <vector 2D> ';';
```

High-level Primitives:

```
<filename>       ::= "Filename" <assign op> <string> ';';
<vector 2D>      ::= '(' <literal> ',' <literal> ')' ';';
```

LIBRARIAN INTERFACE



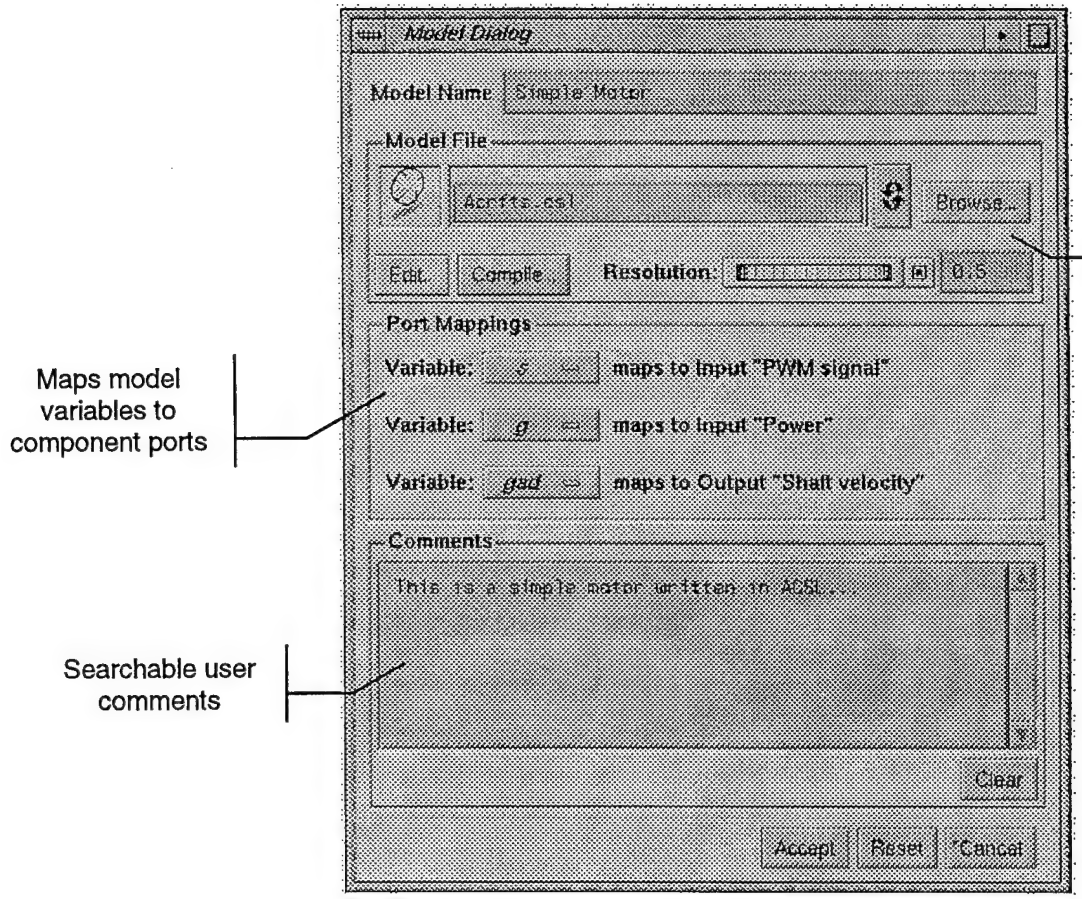
Port definitions

Icon with color-coded ports

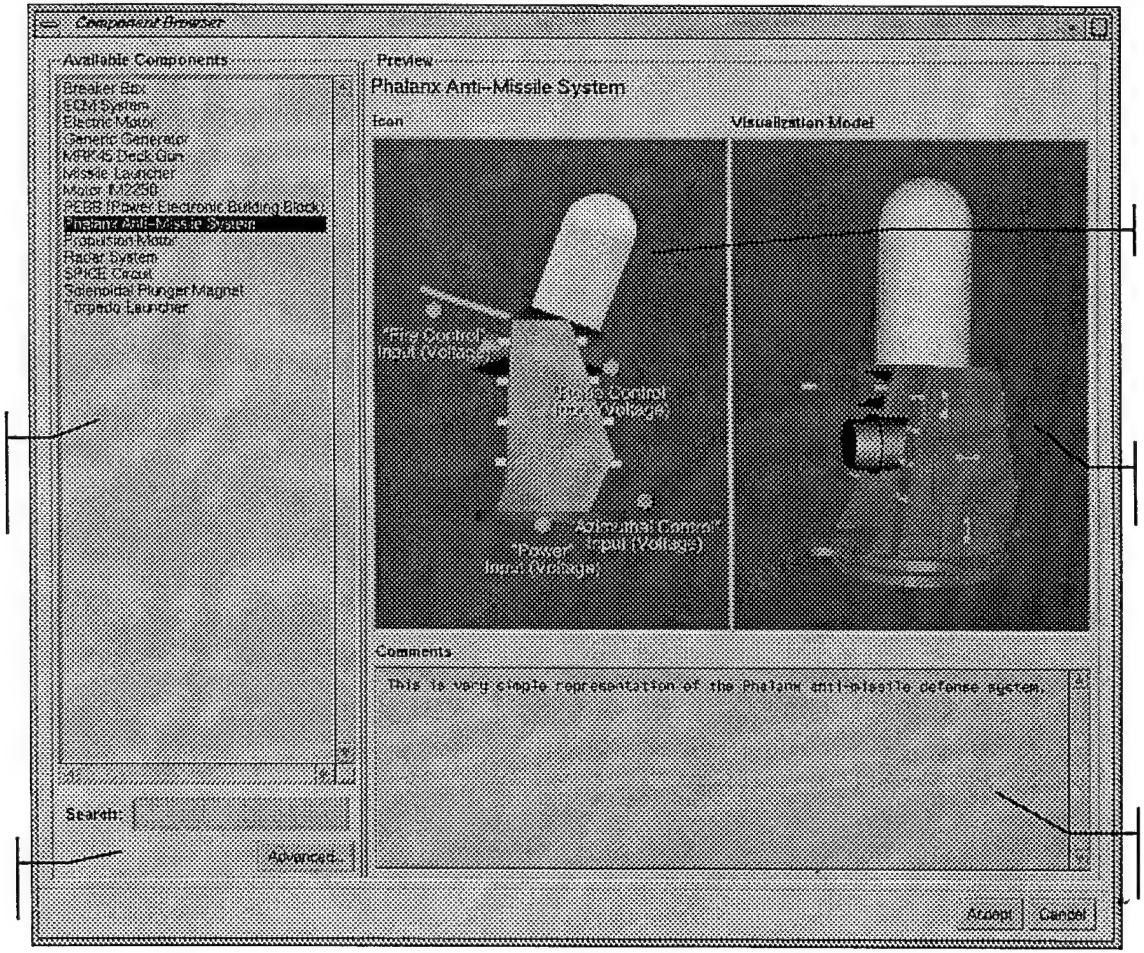
Searchable user comments

One or more representative models

MODEL EDITOR

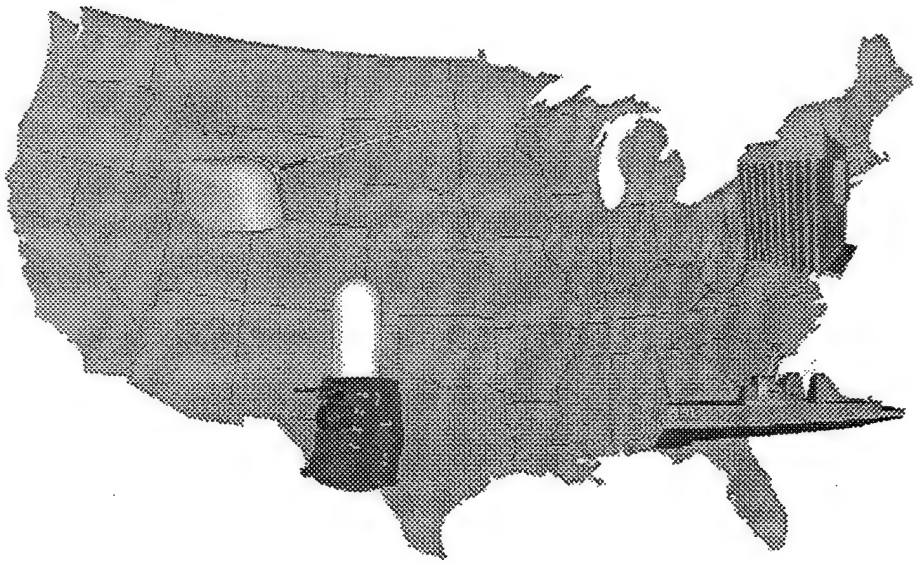


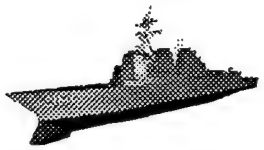
COMPONENT BROWSER



DISTRIBUTED MODEL LIBRARY

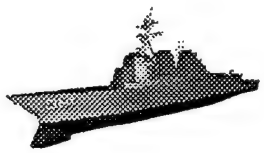
- Models may be located on any computer accessible by the Internet
- Model searches may be local, national, or world-wide in scope





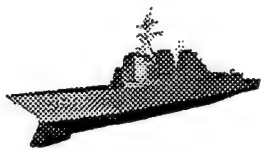
SEARCH SYNTAX

- Search dialog with sophisticated search of component database via web-like search syntax, customized with VTB keywords.
- Example queries:
  - motor **AND ports**=3
  - **model**=**acsl AND** electromagnetic **AND** gun
  - **model**=**spice AND ports**>5



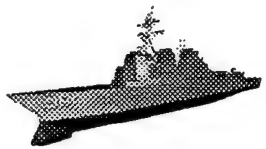
VTB SYSTEM INTERACTION

- Network manager
- Component library
- Translators
- Model library



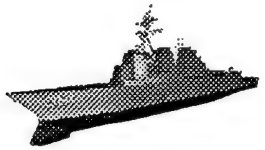
TRANSLATOR INTERACTION

- Extracting symbol table information from a model file
- Embedding backplane information into a model file
- Translation of model into SIL (Solver Interface Language)



NETWORK MANAGER INTERACTION

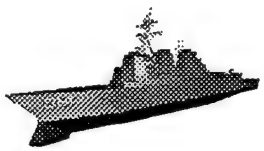
- Automatically parse components
- Provide all component information, including SIL code
- Routines for drawing icons, diagrams, and labels
- Drawing and animation component 3-D models



DIRECTORY ABSTRACTION

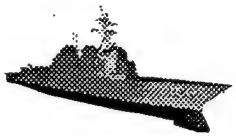
- CVTBPaths class provides abstraction of directory structure via calls
  - char *GetPath(int PathType);
  - char *GetFilename(int PathType, char *);
- PathType can be any of

Location	Image	Action	Creature	DXF
IV	Network	Sounds	Help	XPM
Binary	Acsl	SIL	Component	Icon
Model	Saber	Spice	Doc	



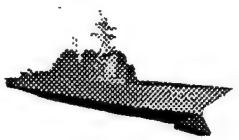
CONCLUSION

- Librarian purpose
- Component definition
- User interfaces
- Distributed library
- VTB system interaction



VIRTUAL TEST BED

Main Program

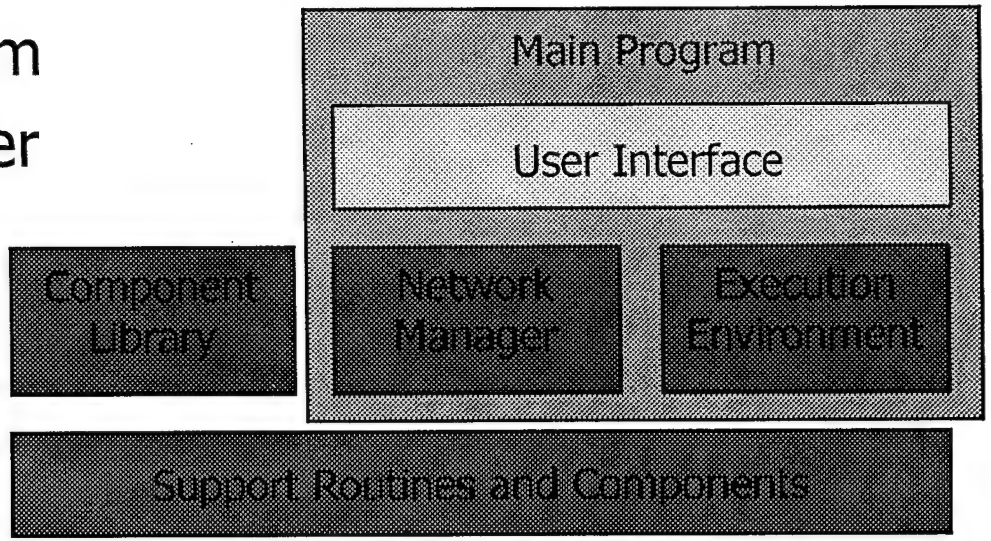


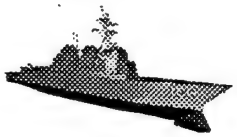
THE VTB MAIN PROGRAM

- Place in software architecture
- Interface layout
- Implementation
- Portability
- Current status and immediate future

WHERE DOES IT FIT?

- Main interface
- Network manager
- Visualization system
- Simulation manager





INTERFACE LAYOUT

MAIN INTERFACE

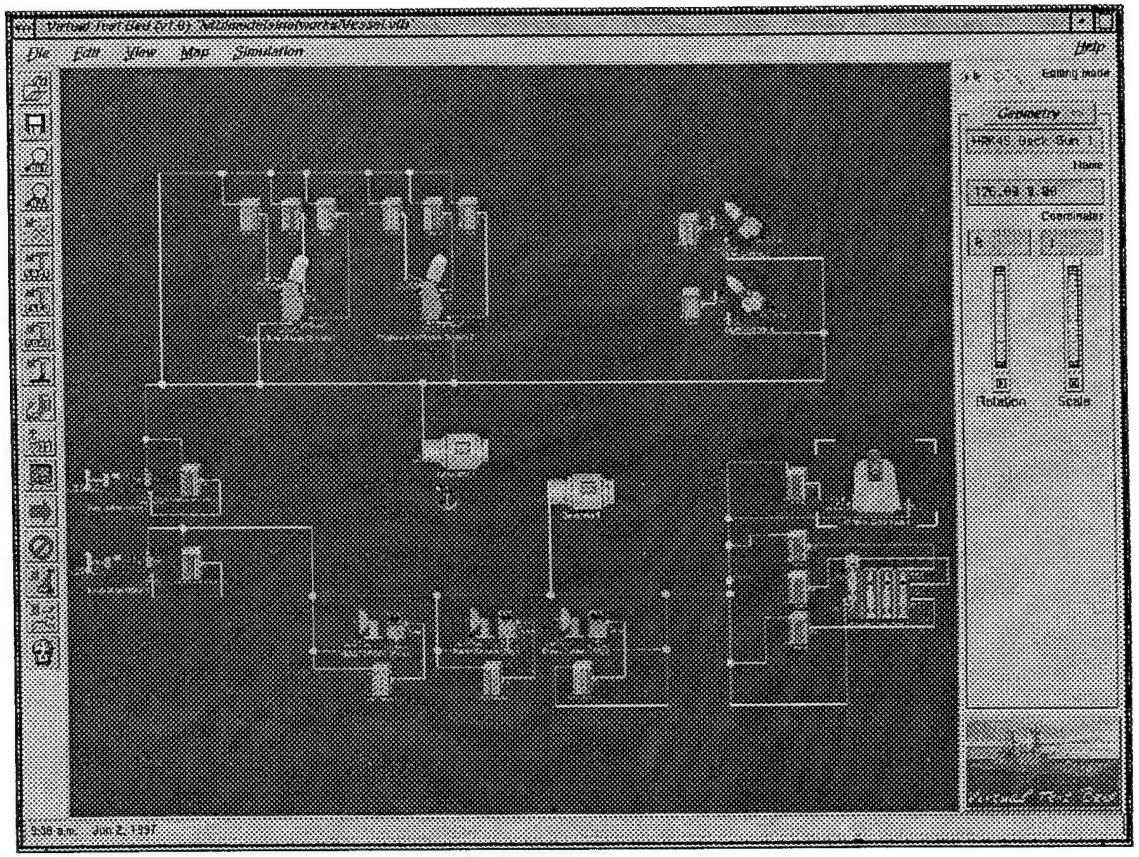
Pulldown menus

OpenGL schematic editing

Customizable toolbar

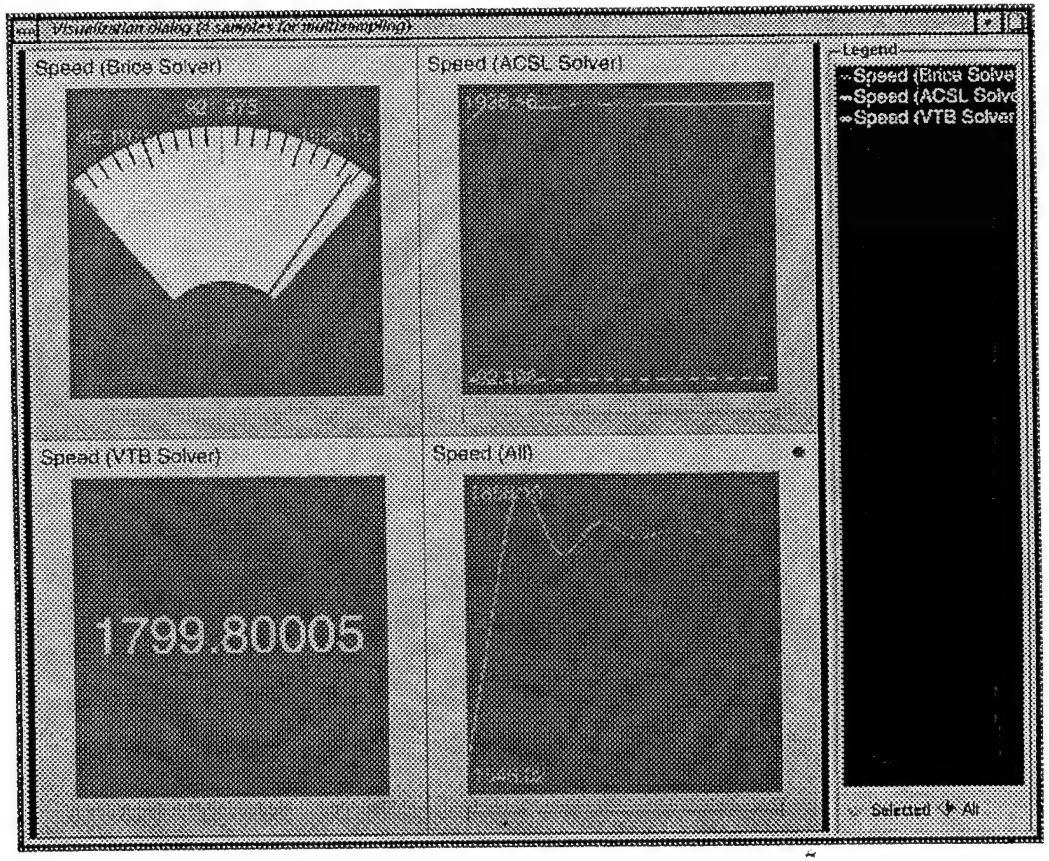
Information/editing region

Status bar



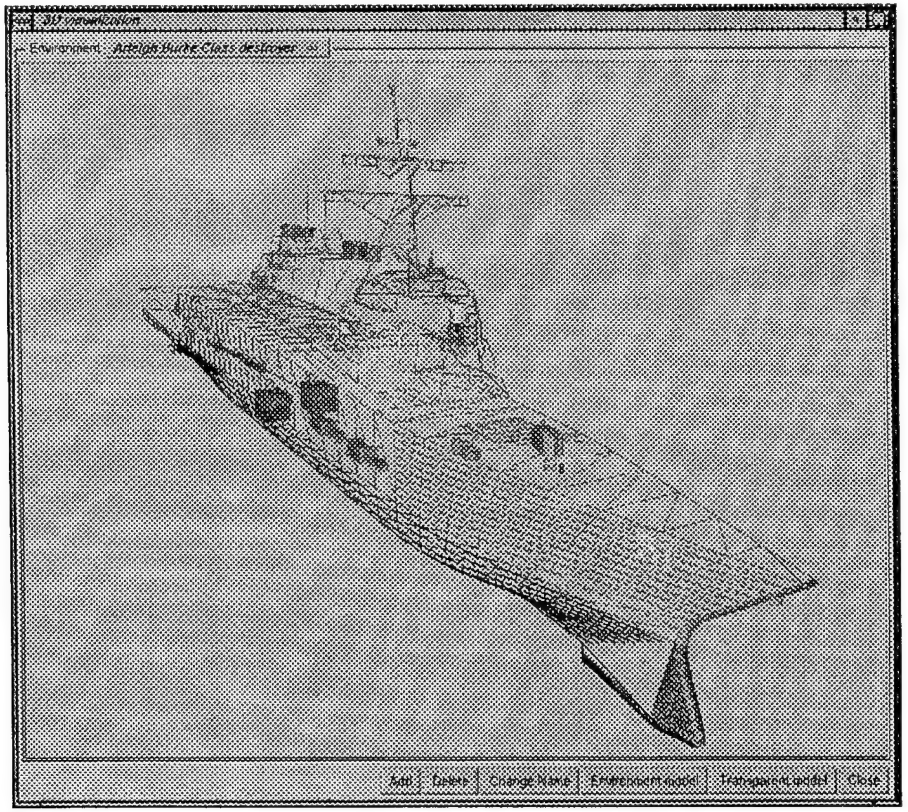
DATA VISUALIZATION

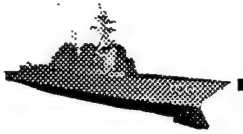
- Contains all meters
- Dials, plots, and digital indicators
- Legend
- Auto-ranging plots and dials
- Overlay capability



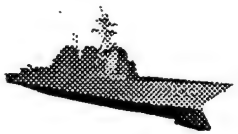
3D VISUALIZATION

- Multiple 3D environments
- Interactive, 3D manipulators
- 3D component animation
- Only the tip of the iceberg
  - Cable placement and length calculations
  - Arbitrary clipping planes for cut-away views
  - Easy plug in for advanced visualization





IMPLEMENTATION



LANGUAGE - "C++"

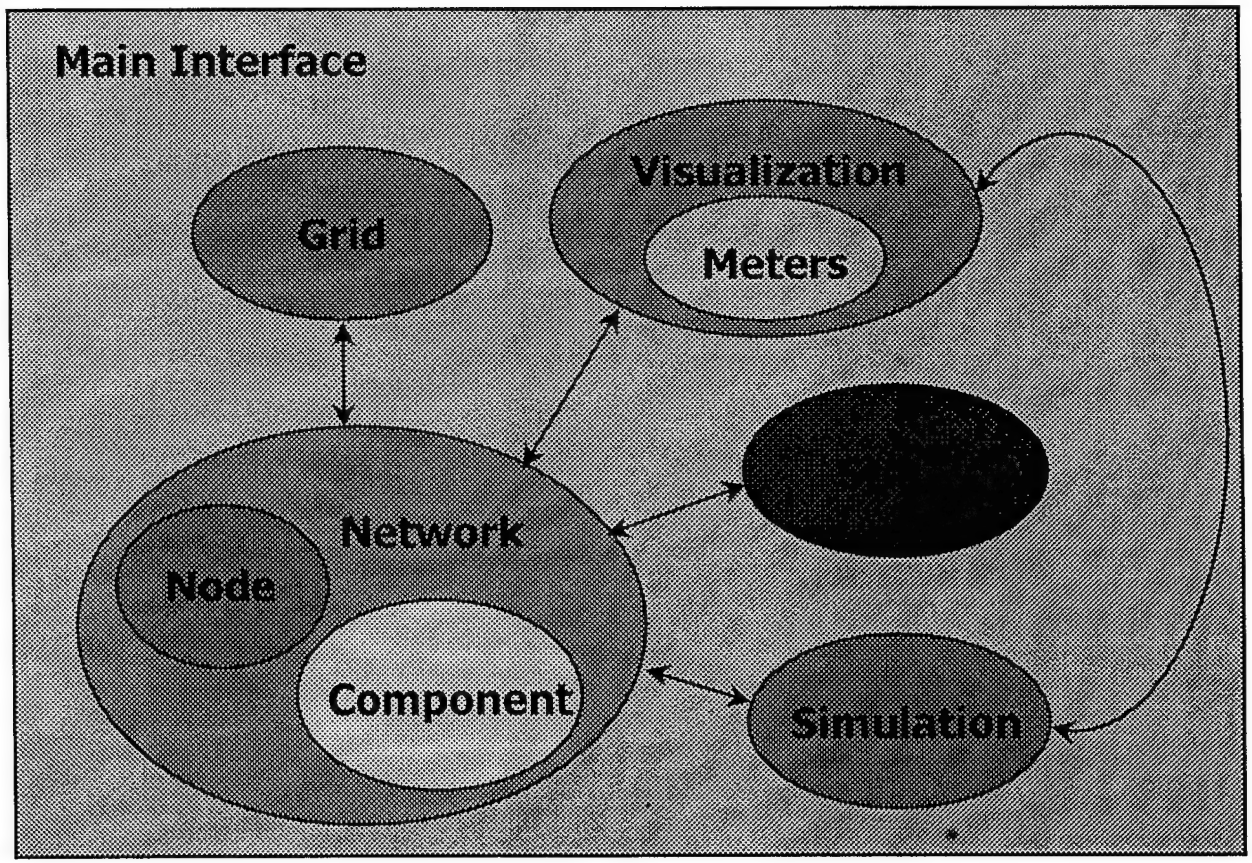
- Problem is naturally object-oriented
- Encapsulation and inheritance
- Target platform is dominated by "C++"
- Allow reuse of legacy "C" code
- Access to OpenGL

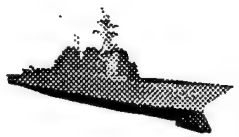
CLASS ORGANIZATION

Class

Interface

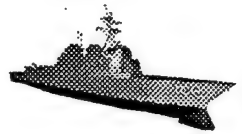
Communication





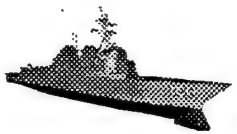
GRID CLASS

- Units information
- View information
  - Current and previous views
  - Grid and snap density
- View operations
  - zooming, panning
  - aspect ratio
- Coordinate abstraction
  - Cursor location (snap)
  - Single pixel size
- OpenGL drawing routines



STATES CLASS

- Stores all interface states
  - Snap
  - Editing mode (component, line)
  - Animation
  - All future states
- Separate from the main interface for portability reasons

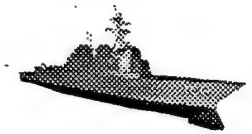


NETWORK CLASS

- Component list
- Node list
- Network manager APIs
  - PickItems(CPnt &pnt, int multiple);
  - AppendVertex(CPnt &pnt);
  - Delete( ), Cut( ), Copy( ), Paste(CPnt &pnt);
  - AddPendingComponent(CPnt &pnt);
  - Overloaded inserter for file I/O.
- OpenGL drawing routines

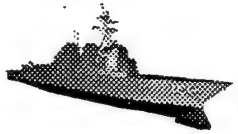
VTB NETWORK FILE

```
Environments
{
"Arleigh Burke Class Destroyer" "arleigh.o.iv" "arleigh.trans.iv"
                    [1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1]
"New environment" "spruance.o.iv"
                    [1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1]
}

Components
{
"Generic Generator 1" "generator.cmp" [t(46 14 0) r(0, 0 0 1) s(1 1 1)]
                    {
                    [1 0 0 0 0 1 0 0 0 0 1 0 1.04e+03 939 47.7 1]
                    [1 0 0 0 0 1 0 0 0 0 1 0 5.77e+03 124 -1.16e+03 1]
                    }
"MRK45 Deck Gun 1" "deck_gun.cmp" [t(0 16 0) r(0, 0 0 1) s(1 1 1)]
                    {
                    [1 0 0 0 0 1 0 0 0 0 1 0 777 201 440 1]
                    [1 0 0 0 0 1 0 0 0 0 1 0 5.77e+03 124 -1.16e+03 1]
                    }
"PEBB 1" "pebb.cmp" [t(-20 12 0) r(0, 0 0 1) s(1 1 1)]
                    {
                    [1 0 0 0 0 1 0 0 0 0 1 0 1.04e+03 6.64e+03 2.38e+03 1]
                    [1 0 0 0 0 1 0 0 0 0 1 0 5.77e+03 124 -1.16e+03 1]
                    }
}

Node 1
{
("PEBB 1" "Pebb Output") (-9.47 11.9);
(-9.47 11.9) ("MRK45 Deck Gun 1" "Control");
}

Node 2
{
("Generic Generator 1" "Power") (26 13.9);
(26 13.9) (26 -2);
(-32 -2) (14 -2);
(-32 15.1) ("PEBB 1" "Input 1");
("MRK45 Deck Gun 1" "Power") (14 10.9);
}

Meters
{
entries
                    {
                    "Generic Generator 1" "x" (1 0 0);
                    "Generic Generator 1" "y" (0 1 0);
                    "Generic Generator 1" "z" (0 1 1);
                    "MRK45 Deck Gun 1" "x" (0 0 1);
                    "MRK45 Deck Gun 1" "y" (1 1 0);
                    "MRK45 Deck Gun 1" "z" (1 0 1);
                    }
"Logic waveforms"(plot)(dynamic)(-10 10)
                    {
                    (1 0 0);
                    (0 1 0);
                    (0 1 1);
                    }
"Power meters"(digital)(dynamic)(-10 10)
                    {
                    (0 0 1);
                    (1 1 0);
                                        }
}
```



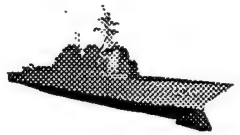
VTB NETWORK FILE BNF

<VTB Network File>	→	[<3D Environments>] [<Component list>] {<Node>} [<Meter List>]
<3D Environments>	→	**Environments** '{' {<Environment>} '}'
<Environment>	→	<User Id> <User Id> [<User Id>] < 3D Transform>
<Component list>	→	**Components** '{' {<Component>} '}'
<Component>	→	<User Id> <User Id> <2D Transform> '{' {<3D Transform>} '}'
<Node>	→	**Node** <Literal> '{' {<Connection>} '}'
<Connection>	→	<Vertex> <Vertex> ';'
<Vertex>	→	<Coordinate> | <Port>
<Coordinate>	→	'(' <Literal> <Literal> ')'
<Port>	→	'(' <User Id> <User Id> ')'
<Meter List>	→	**Meters** '{' <Entry List> {<Meter>} '}'
<Entry List>	→	**entries** '{' {<Meter Entry>} '}'
<Meter Entry>	→	<User Id> <User Id> <Color> ';'
<Meter>	→	<User Id> <Indicator type> <Range type> <Manual range> '{' {<Color>} '}'
<Indicator type>	→	'(' **plot** | **dial** | **digital** ')"
<Range type>	→	'(' **manual** | **dynamic** | **framing** ')'
<Manual range>	→	'(' <Literal> <Literal> ')'



COMPONENT CLASS

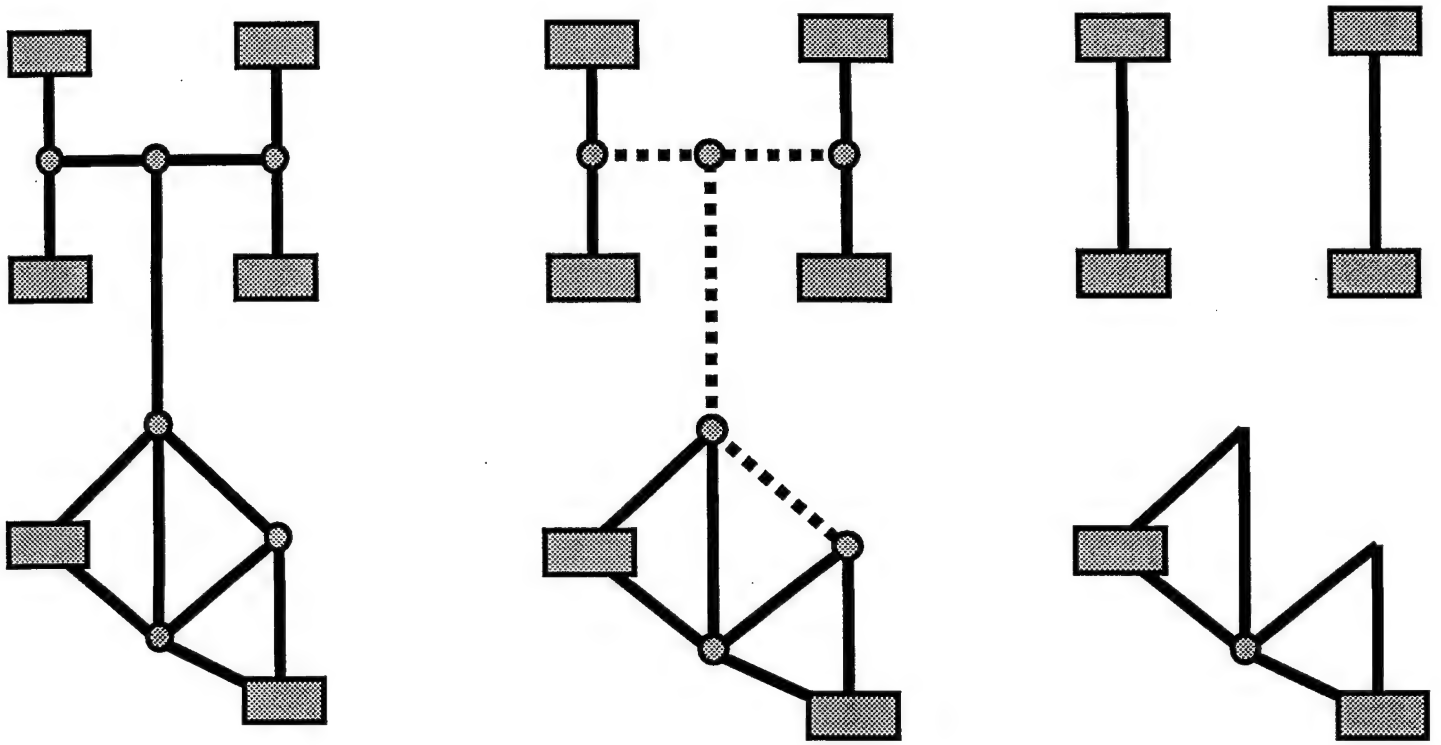
- Inherits component library API's from Component Librarian base class
- Stores additional information required by the interface
  - Geometric transformations (2D & 3D)
  - User defined component names (i.e., generator 4a)



NODE CLASS

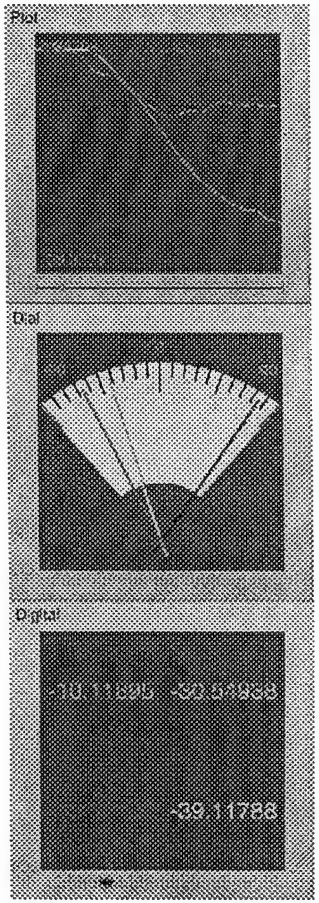
- Provides component connection API's to the network manager
  - AppendVertex(CPnt &pnt, int elbow);
  - AppendVertex(CComponent *comp, CPort *port);
  - RemoveConnection(Cvertex *v1, Cvertex *v2);
- Performs type checking
- Uses directed graphs to store line vertices and connectivity information

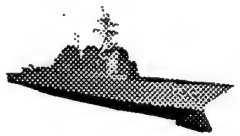
NODE CLASS (continued . . .)



METER CLASS

- Data inspection
- Plots, dials, and digital indicators
- Auto-ranging plots and dials
- Domain view control for plots
- Multiple variables per meter

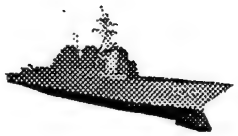




SIMULATION CLASS

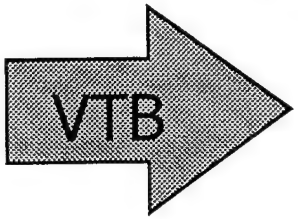
- Controls simulation execution
- Responsible for mission script implementation

PORTABILITY



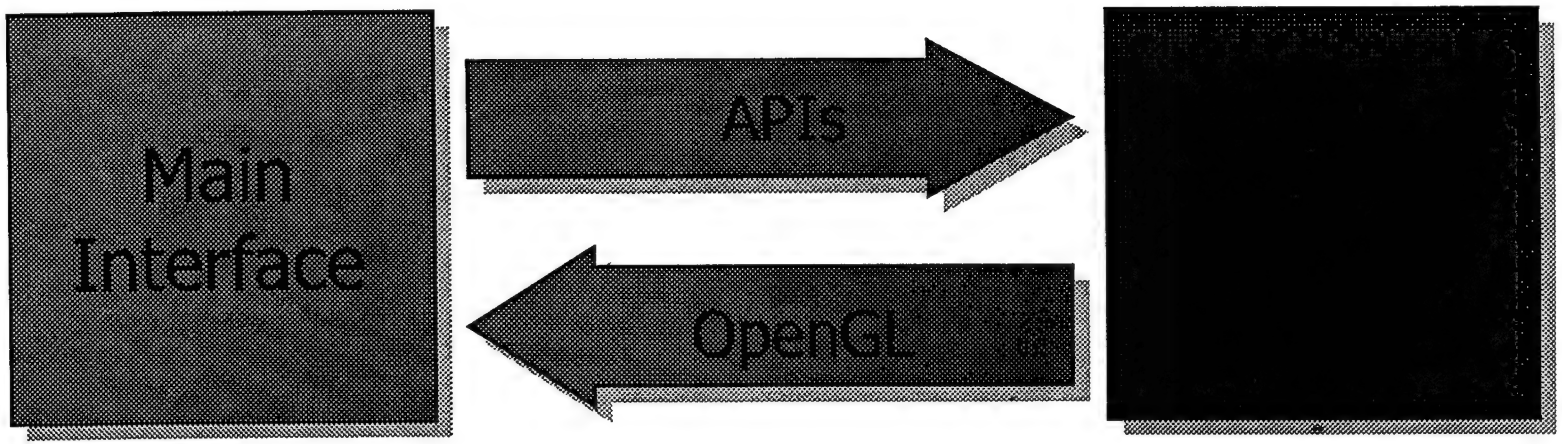
PORTING STRATEGIES

- Run an X-server on the NT machine
  - pro - Very little code to port
  - con - Requires NT users to run an X-server
  - con - Lose advantages provided by MFC
- Develop cross-platform interface language
  - pro - Nothing to port
  - con - Extremely time consuming
  - con - Lose advantages provided by MFC
- Careful code organization
  - pro - All platforms have native interfaces
  - pro - Enforces "portability aware" coding practices
  - con - X-Windows specific code must be rewritten

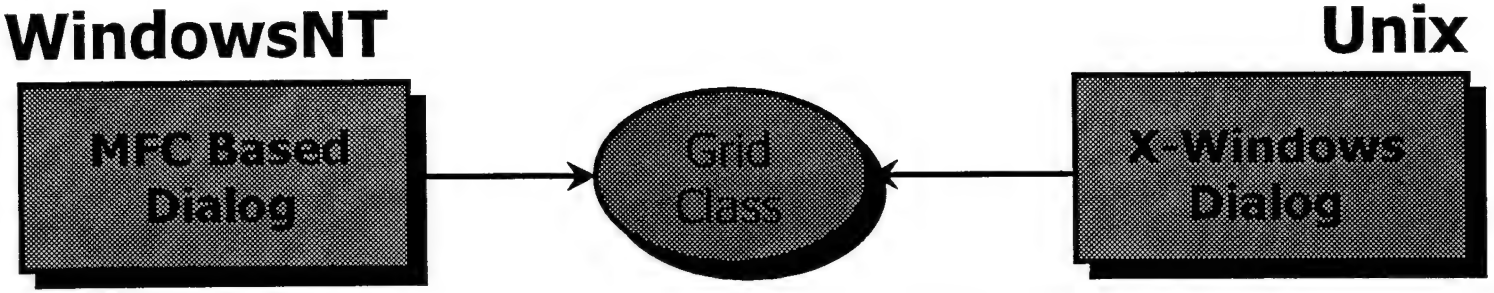


ORGANIZATION

- Use of OpenGL wherever possible
- Visual programming
- No major internal class will contain interface specific code



DIALOG EXAMPLE




```
void   CGridDialog::OnFeet( )
{
       CVTBDocument *pDoc = CVTBDocument::GetDocument( );

       pDoc->m_grid.SetUnits(GRID_FEET);
}
void   CGridDialog::OnInches( )
{
       CVTBDocument *pDoc = CVTBDocument::GetDocument( );

       pDoc->m_grid.SetUnits(GRID_INCHES);
}
```

```
static void   UnitsCB(Widget w, int option)
{
switch (option)
       {
       case 0:
              grid.SetUnits(GRID_INCHES);
              break;
       case 1:
              grid.SetUnits(GRID_FEET);
              break;
       default:
              DEFAULT("UnitsCB (grid dialog)");
       }
}
```

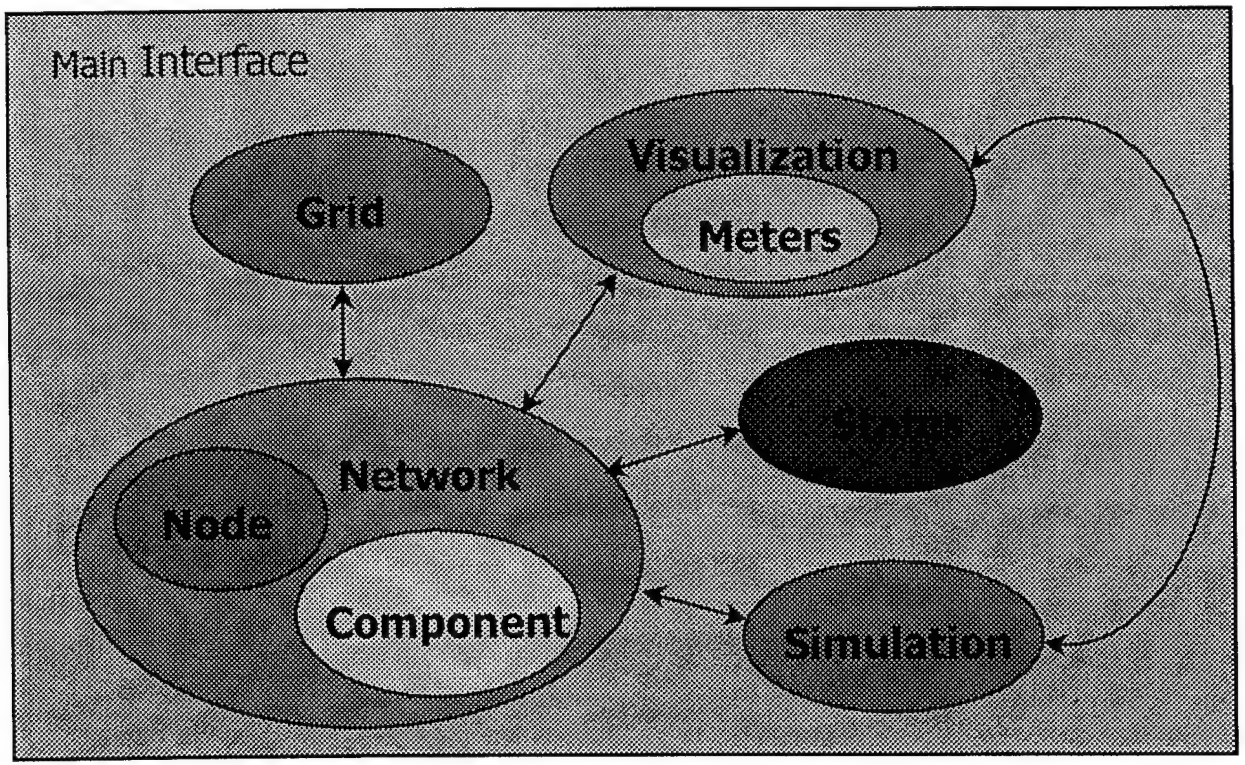
PORTABILITY ESTIMATES

Network
100%

Component
100%

Node
100%

Visualization
100%



Meters
100%

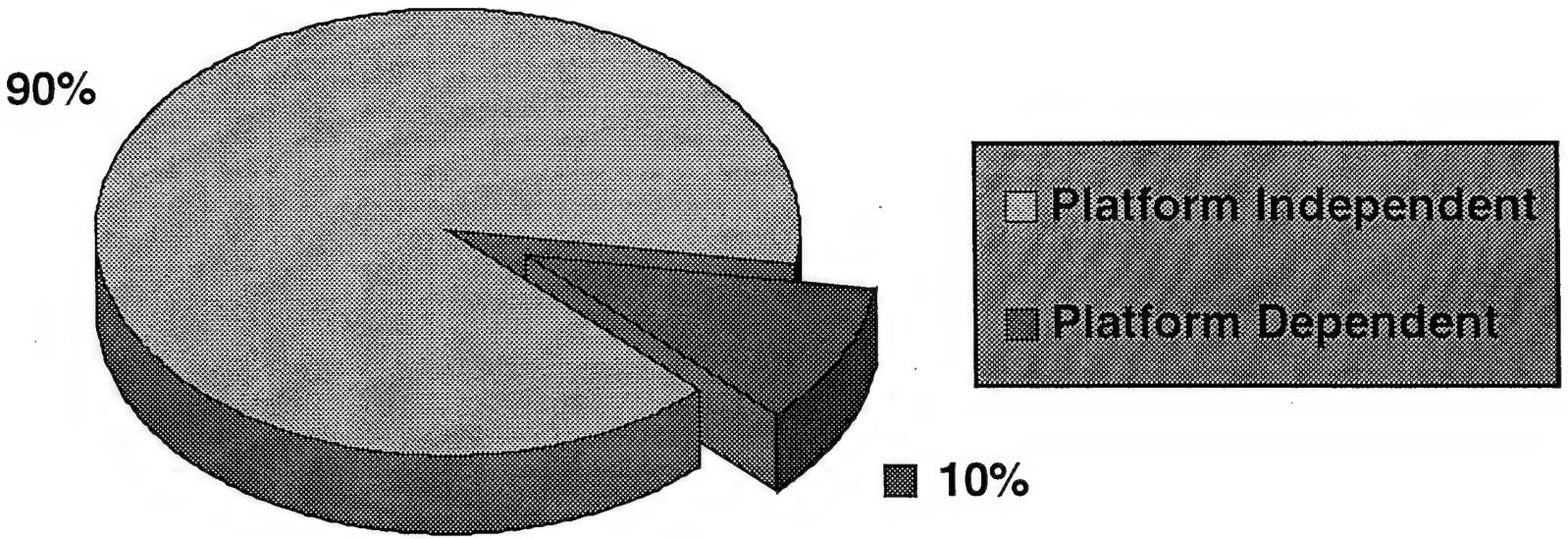
Simulation
100%

Grid
100%

States
75%

Main Interface
0%

PLATFORM DEPENDENCE



ASSESSMENT AND FUTURE DIRECTIONS

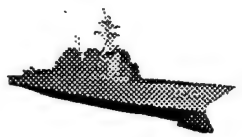
WHERE ARE WE NOW?

Functional Framework for Network Construction

- Component librarian and main program have been integrated
- The GUI framework has been implemented
- Network construction GUI is complete
- Line drawing and component connection mechanisms are in place
- Network files can be saved and retrieved for subsequent editing

First Pass at Solver Integration and Data Visualization

- Solver results can be loaded for use with a network file
- Meters provide a versatile method for data inspection
- 3D animation based on simulation variables



WHERE ARE WE GOING?

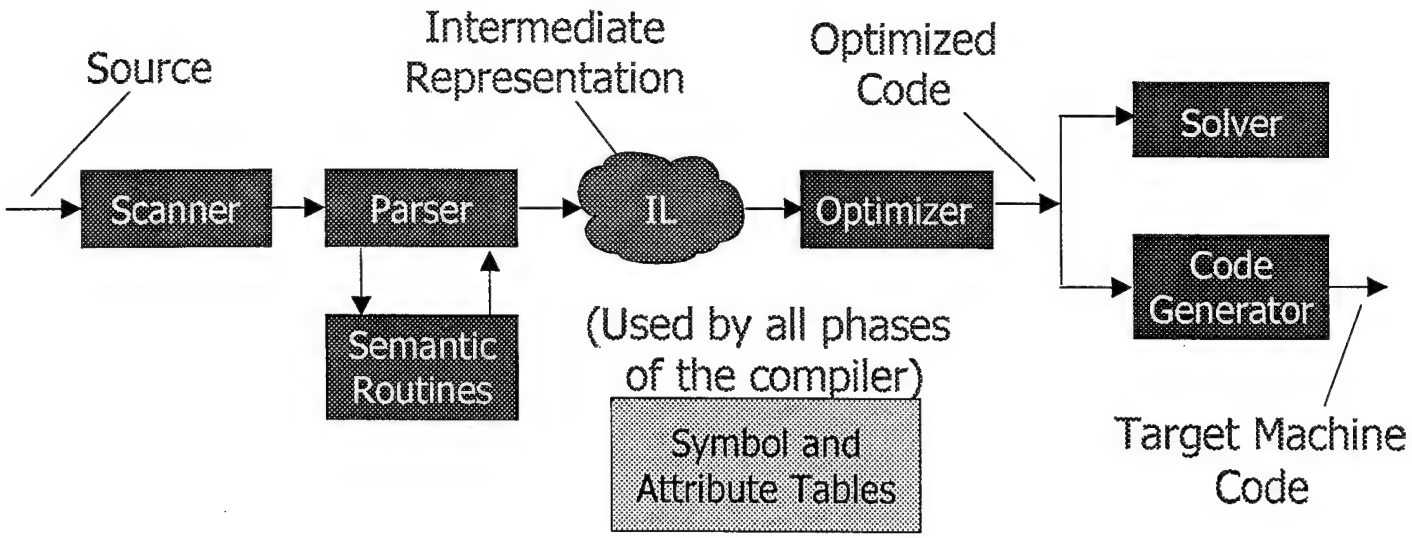
- Advanced visualization techniques
- Increased meter functionality
- Solver Interface Language generation
- "Mission" based simulation control

TRANSLATORS, SOLVER INPUT LANGUAGE, AND SOLVER

- Overview
- Translators
- Solver
- Solver Input Language (**SIL**) ⇐ Focus
- Solver Backplane

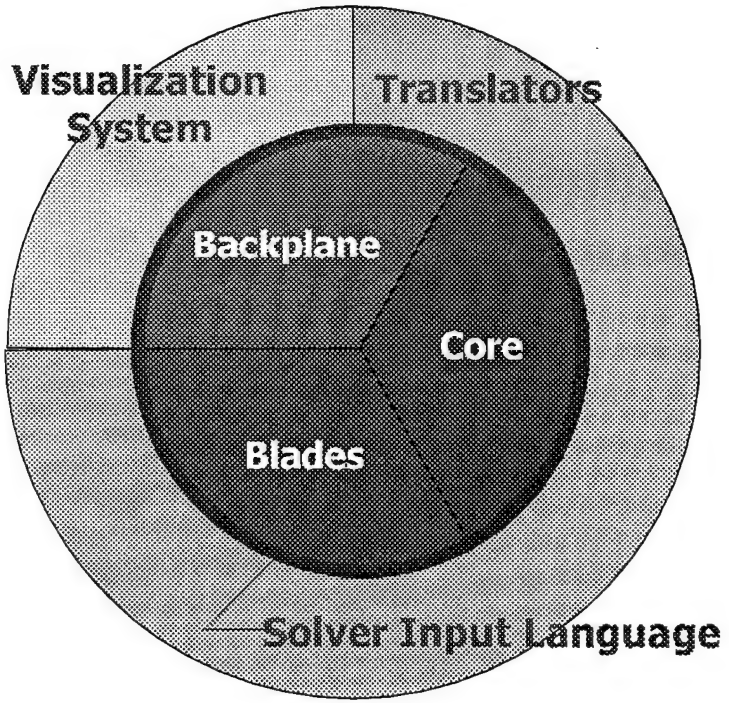
OVERVIEW - 1

Compiler structure applied to VTB



OVERVIEW - 2

- SIL consists of 1) core, 2) backplane, and 3) the blades.
- SIL abstracts both the translators and the visualization system from the execution environment.
- Backplane provides inter-object communication.
- Blades provide language-specific sevices.



TRANSLATORS

- Preprocessor
- Lexical Scanner
- VTB Translator Front Ends (Parser)
- VTB Translator Back End (Code Generator)
- Optimizer

PRE-PROCESSOR

A Language-Specific Preprocessor Is Implemented

- The User-Specified Input File Is Preprocessed to a New File With the Existing Name with the ".pre" Suffix ppended
- Examples of the Types of Features Supported
  - INCLUDE Directives
  - Macro Directives
  - External Language indings (e.g., CSL/FORTR N Subroutine indings)
- T Translator-Specific Directives are Embedded in the ".pre" File Such that Error Line Numbers Continue to be Specific and appropriate
  - Likewise, Great Care Is Taken to Preserve the User Look-and-Feel of Code
- The Preprocessor Is Implemented from the Ground-Up in C++

SCANNER

- Implemented using Lex/Flex with C/C++ supporting subroutines
- Uses a stack and link-time bindings to communicate with parser
- Handles all lexical issues, such as white space, comments, case sensitivity, etc.

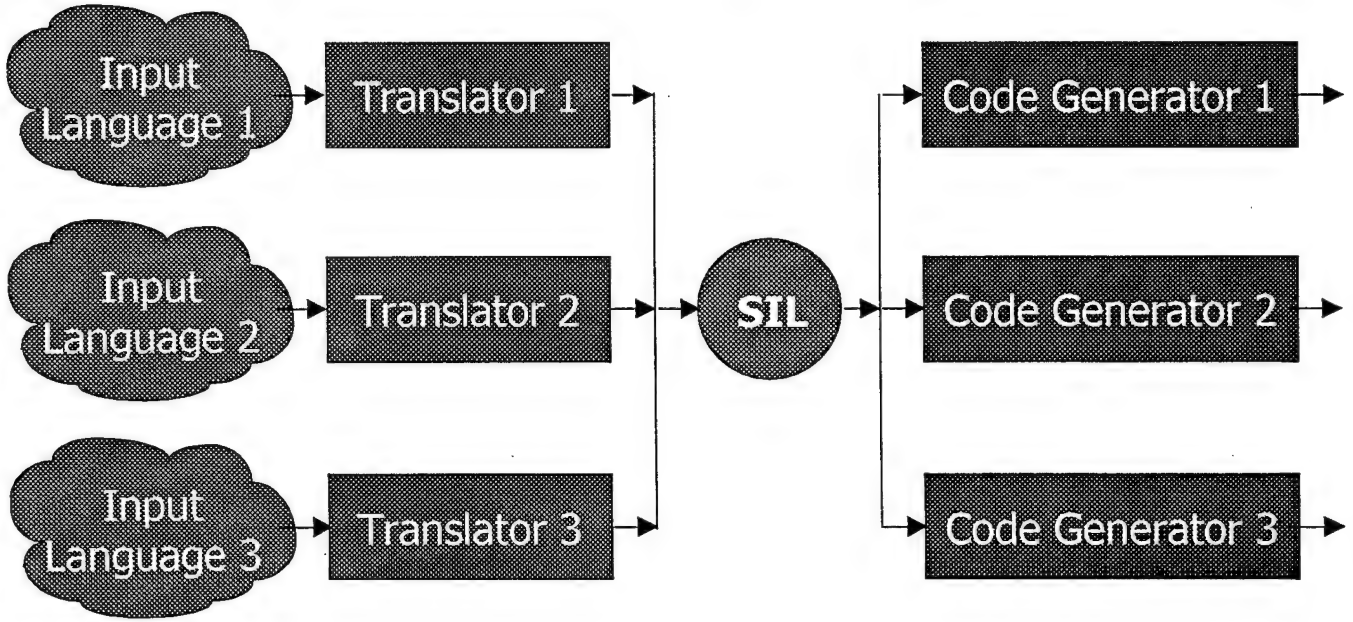
PARSER

- Implemented in Yacc, using an LR grammar
- Parser is organized as a set of BNF-like rules
- Semantic routines are attached C/C++ methods
- Yacc limitation of LALR(1) is handled by semantic routines and by scanner.

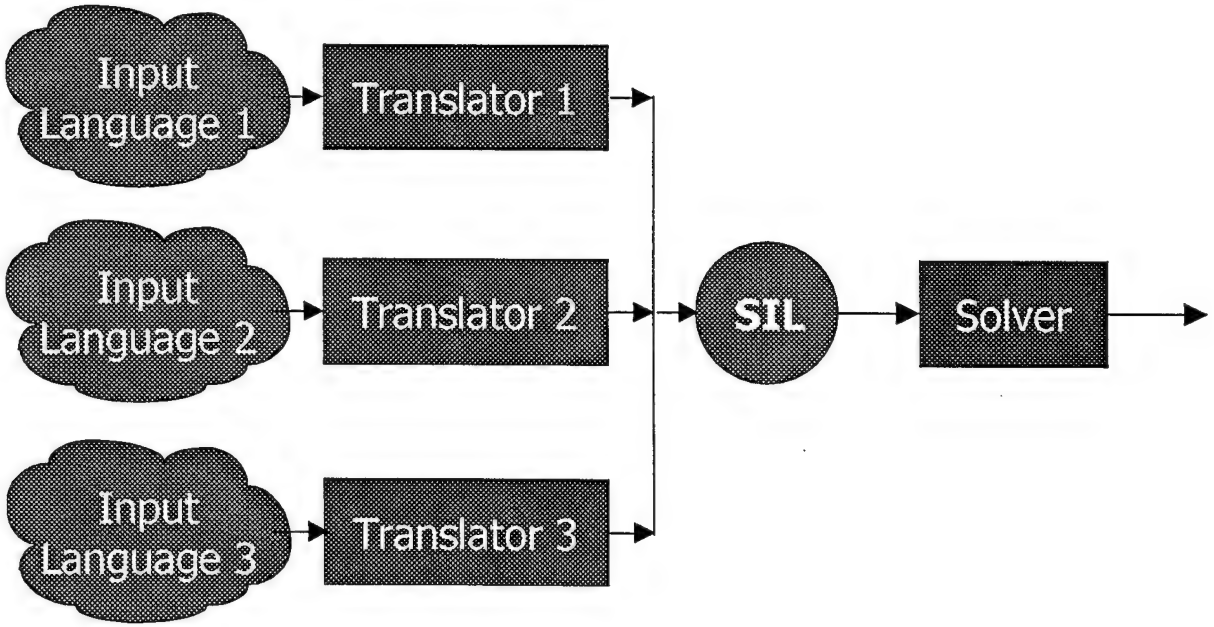
PARSER LIMITATIONS

- ACSL/FORTRAN reserved words may not be used as identifiers (intentional restriction)
- Lack of a formal definitions for most languages, such as ACSL, means that parser is ad hoc. Proof of correctness is via running hundreds of ACSL programs.

M + N ARCHITECTURE



CURRENT ARCHITECTURE



SOLVER

- The intermediate language, and not the solver, defines the execution environment.
- Micro-code-like implementation architecture, consisting of a number of independent engines.
- Solver directs each tuple to the correct engine.
- Current solver reads ASCII names for clarity.

INTERMEDIATE LANGUAGE

- Standard "tuple" format
- Storage management and control flow
- SIL assembler will be done later
- Flat vs modular models

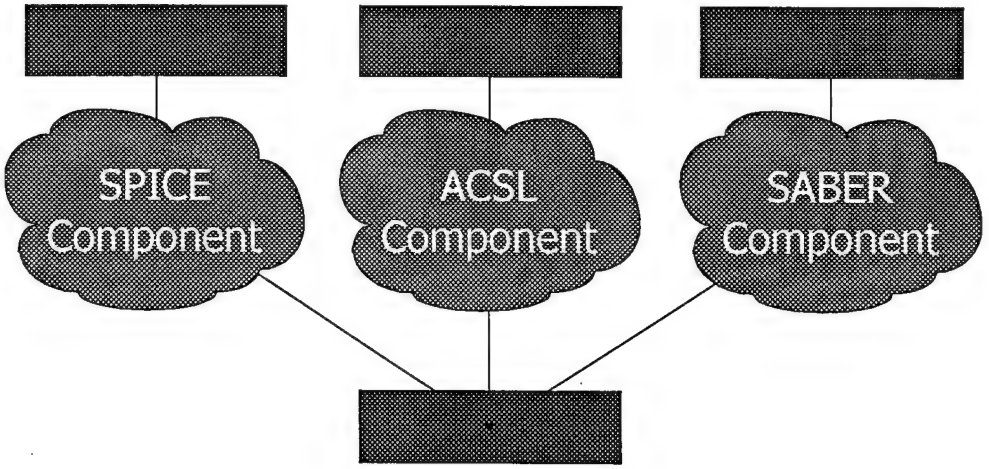
SIL INSTRUCTIONS

SIL supports two types of instructions, tuples and run-time methods

- Tuples ⟹ simple lists of the type <operator, operand1, operand2, .. operand#>
- Run-time methods are equivalent to system calls in a standard environment

FLAT vs MODULAR

MODULAR APPROACH: Each component executes as separate, "modular" sequence of instructions.



FLAT MODEL: All simulations execute in a single, flat instruction space.

SIL CORE

- Consists of those tuples and run-time methods that are common to all supported languages.
- Goal is to maximize the core set to simplify implementation and enhance optimization.

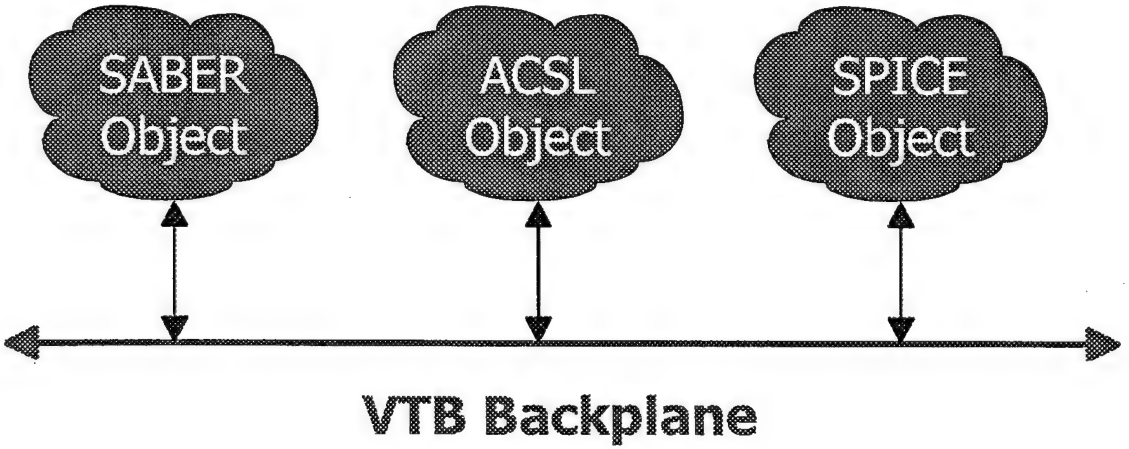
SIL BLADES

- A SIL blade is a standard software module that plugs into the solver and extends its capabilities to handle specific languages
- SIL blade architecture defines methods for adding functionality and increases focus on commonality of the language.
- ACSL CINTERVAL is example blade feature.

EMBEDDED LANGUAGES

Embedded languages, such as FORTRAN in ACSL, are not supported by SIL. This code is shunted off by the translator and a run-time link is built. The SIL "coreCall" instruction is used to run the resulting object.

THE BACKPLANE



The VTB backplane provides the VTB mechanism for inter-object communication.

GRAMMAR FEATURES

TYPE DECLARATIONS - BackplaneCurrent, BackplaneImpedance, BackplaneCapacitance, BackplaneInductance, Backplane oltage, Backplane elocity, BackplaneForce BackplaneAngularAcceleration, , BackplaneAcceleration, Backplane orque, BackplaneAngular elocity, BackplaneImpedance

DOMAIN - BackplaneRange

TYPE MODIFIERS - BackplaneInput, BackplaneOutput, BackplaneInputOutput

STRUCTURE - BackplaneBegin, BackplaneEnd

RUN-TIME SUPPORT - BackplaneRead BackplaneWrite, Backplane race, Backplane ime, BackplanePause, BackplaneSleep, BackplaneResume

ACSL BACKPLANE EXAMPLE

```
Example VTB ACSL program.
!
!BackplaneBegin
!BackplaneInput BackplaneVoltage 0.0..1000.0      Voltage
!BackplaneInput BackplaneCurrent 0.0..500.0       Current
!BackplaneOutput BackplaneTorque                  Torque
!BackplaneEnd
!
PROGRAM A VTB Backplane Example Program
      REAL x, y, z
      INTEGER   Torque
      REAL      Voltage, Current
      .
      .
```

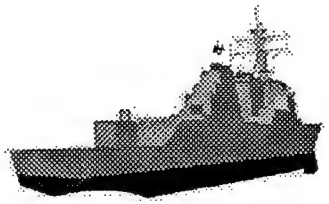
The input variables **Voltage** and **Current**, and the output variable **Torque** can be used externally. The real variables **x**, **y**, and **z** are internal to the object.

OPTIMIZATION

- Most optimization in a typical compiler is done at the intermediate code level.
- The use of an intermediate language means that this can be done with VTB.
- The optimization step would apply to all languages and would be independent of the execution environment.

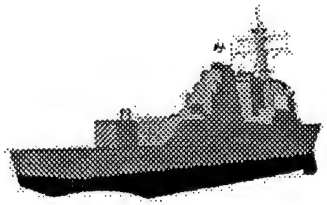
SUMMARY

- Each VTB component is an object and each object can be an actor.
- Most backplane instructions are contained in the object
- The solver/code generator decision is transparent to the translators and the objects.
- SIL will support all standard real-time methods
- Optimization can be done on SIL - all supported languages benefit



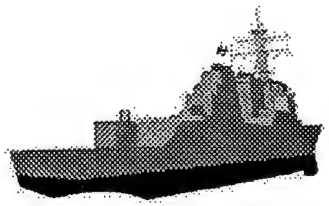
Numerical Methods

- The heart of the VTB will be a discrete-event simulator that drives numerical solution of algebraic and differential equations in the time domain.
- Other functions, such as frequency-domain analysis will ride on top of the time domain simulation.



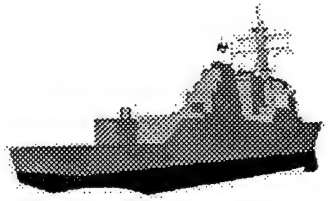
Definition

- The numerical solver computes the future state of the system, step-by-step in the time domain
- The solver is affected by the mission only when the mission changes a model or an input
- The solver coordinates model objects at run time via the backplane



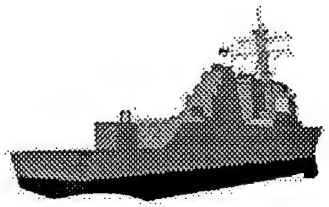
Vision

- Object-oriented solution methods
  - Each object may have its own solver and unique solution methods
- Discrete-event simulation becomes a true real-time system



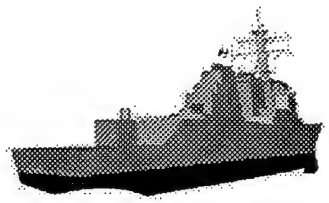
Requirements

- Modular solution space requires the simulation backplane
  - Each subsystem model may run at its own rate, but synchronized to the system time
  - VTB backplane allows inter-model communication
  - Solver interface language supports the backplane



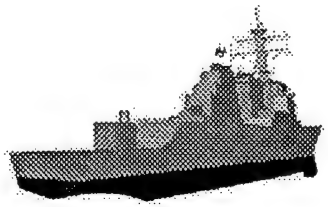
Requirements

- Solver must handle:
  - Initial-value problems (ordinary differential equations)
  - Differential-algebraic equations (differential equations coupled by algebraic constraints)
  - Eventually: boundary-value problems (perhaps coupled with initial-value time-domain solution)



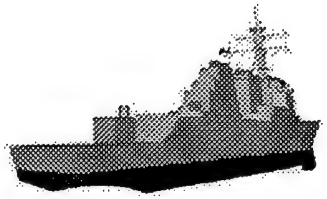
Risks

- Implementation of real-time concurrent processes
  - Visualization system: LOW (similar applications are successful)
  - Model to model interactions: MODERATE (development issue for year two)
  - Automatic negotiation between model objects: HIGH (research issue)



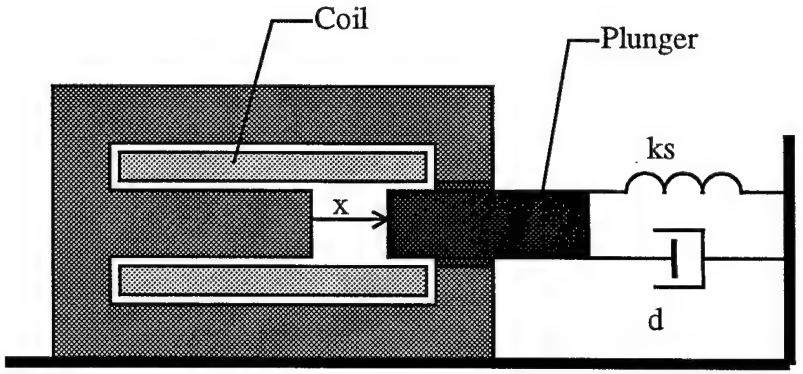
Year One Implementation

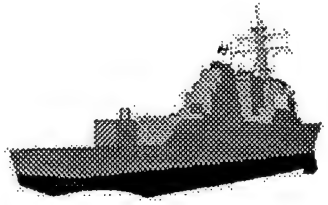
- Flat solution space
- Numerical solution of differential equations by semi-implicit trapezoidal rule algorithm
- Serves as proof of concept
- Two demonstration physical systems

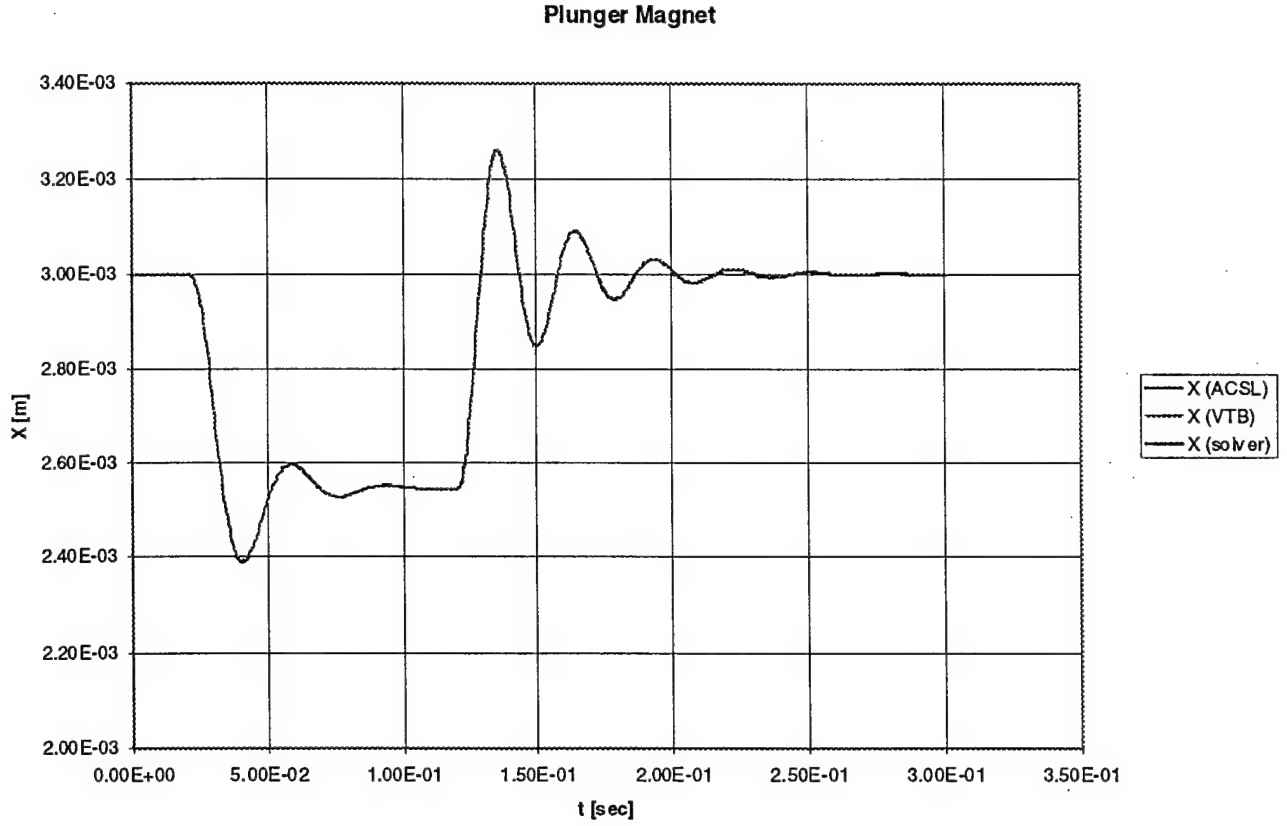


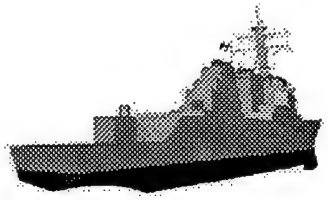
Plunger Magnet

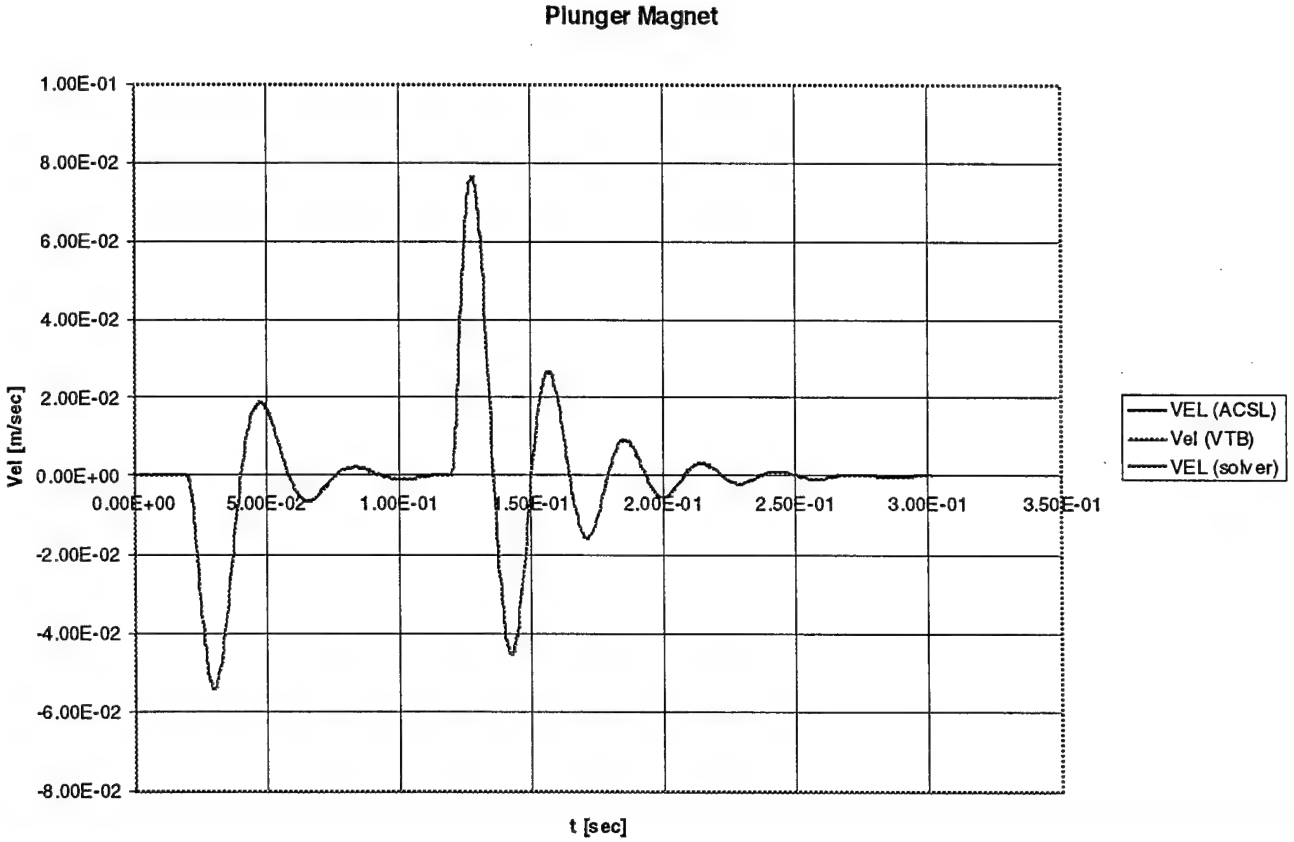
- Cylindrical solenoidal plunger magnet
- Coil excited by a voltage source

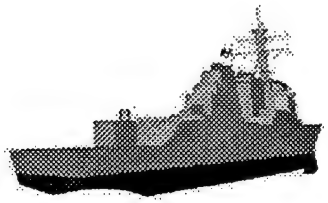





Plunger Magnet
3.40E-03
3.20E-03
3.00E-03
2.80E-03
2.60E-03
2.40E-03
2.20E-03
2.00E-03
X [m]
0.00E+00
5.00E-02
1.00E-01
1.50E-01
2.00E-01
2.50E-01
3.00E-01
3.50E-01
t [sec]
X (ACSL)
X (VTB)
X (solver)

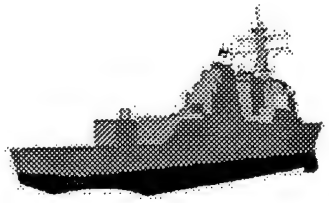



Plunger Magnet
1.00E-01
8.00E-02
6.00E-02
4.00E-02
2.00E-02
0.00E+00
-2.00E-02
-4.00E-02
-6.00E-02
-8.00E-02
Vel [m/sec]
0.00E+00
5.00E-02
1.00E-01
1.50E-01
2.00E-01
2.50E-01
3.00E-01
3.50E-01
t [sec]
VEL (ACSL)
Vel (VTB)
VEL (solver)

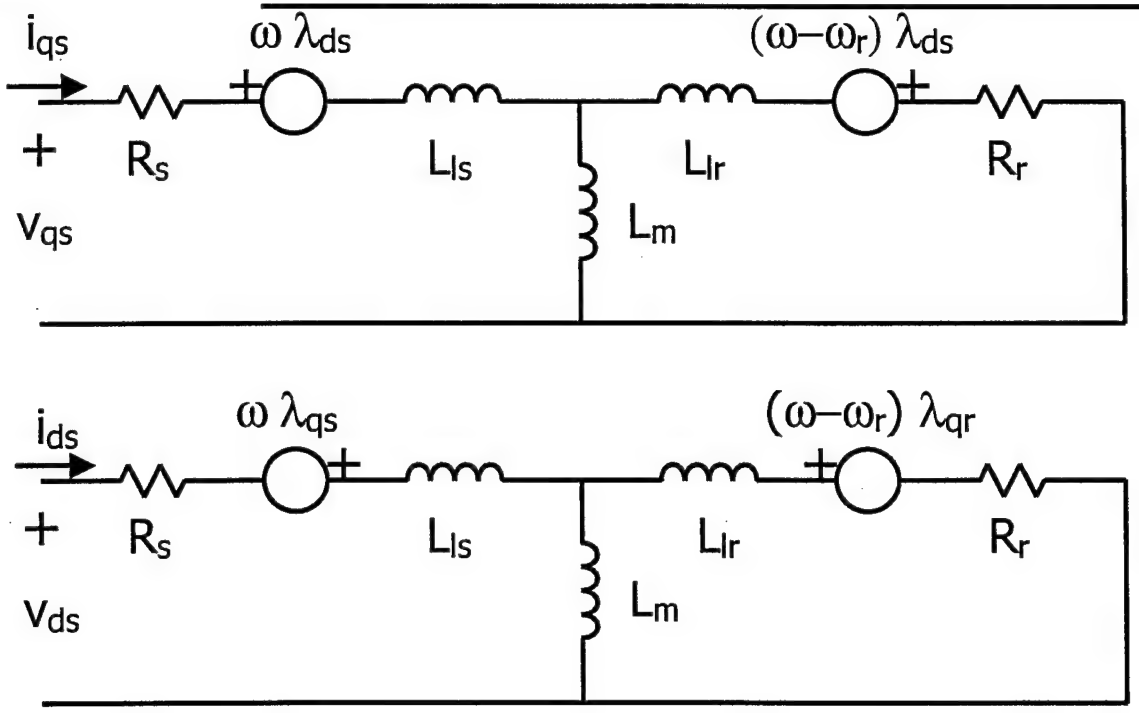


Induction Motor

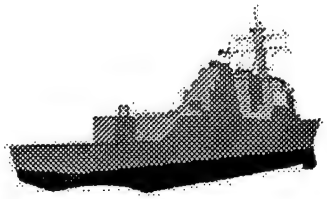
- A 2250 hp, 2300 volt, 3-phase induction motor starts across a stiff voltage source with no shaft load
- The motor is simulated in ACSL and using our solver with excellent agreement

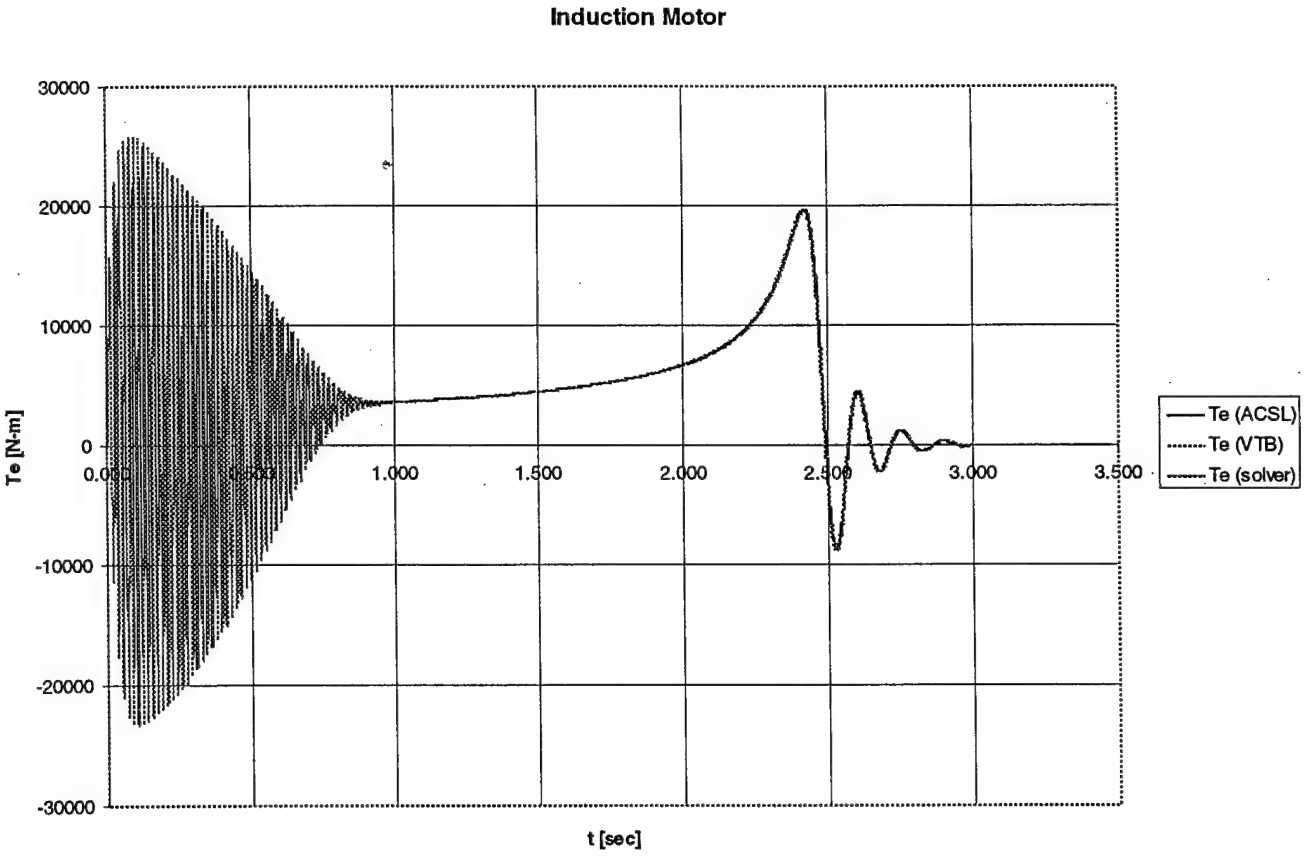


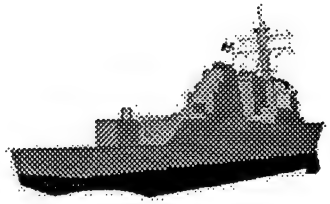
Induction Motor

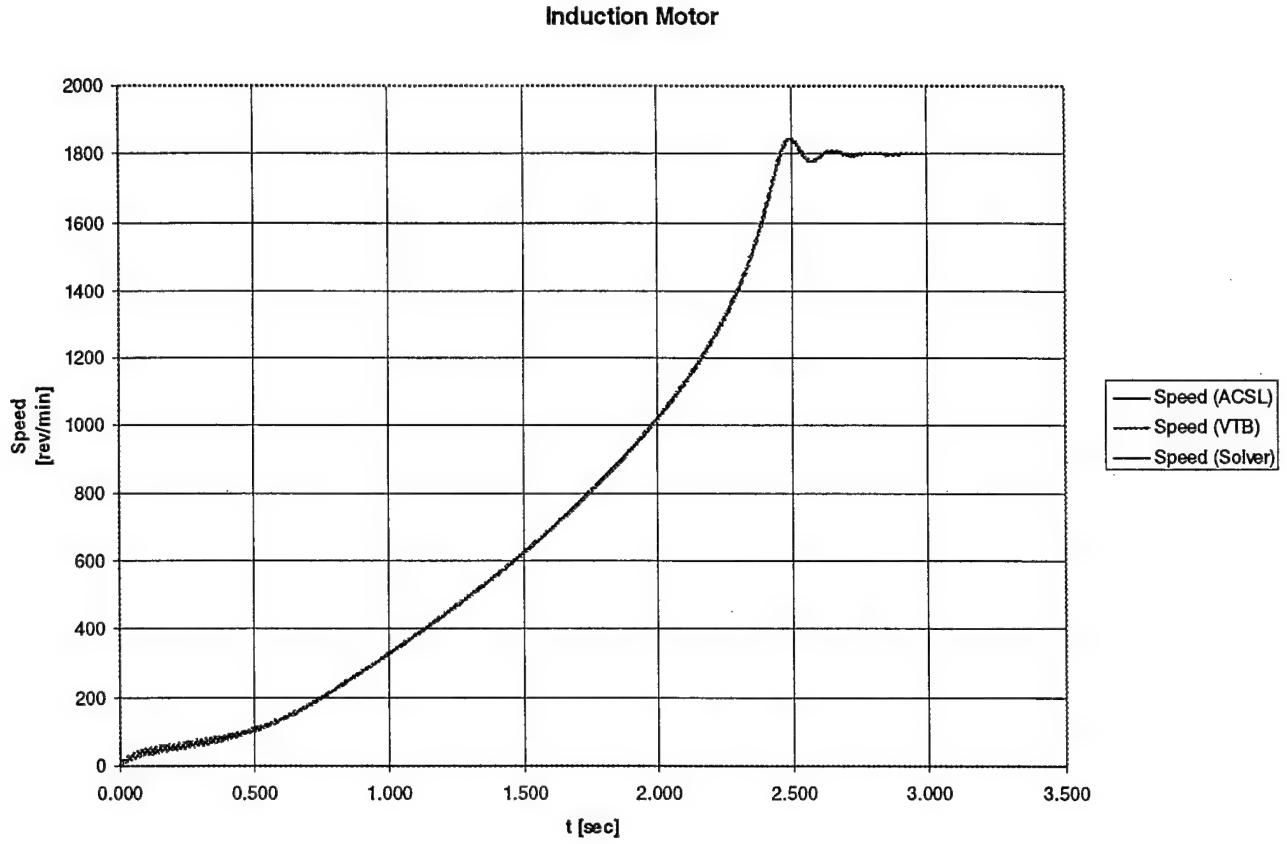


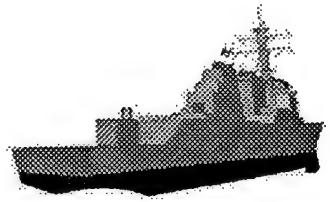
Reference frame rotating at ω electrical rad/sec




Induction Motor
30000
20000
10000
0
-10000
-20000
-30000
Te [N-m]
0.000
1.000
1.500
2.000
2.500
3.000
3.500
t [sec]
Te (ACSL)
Te (VTB)
Te (solver)

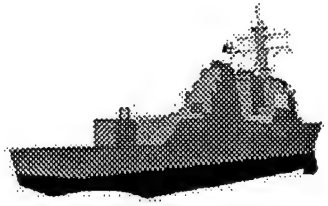



Induction Motor
2000
1800
1600
1400
1200
1000
800
600
400
200
0
Speed [rev/min]
0.000
0.500
1.000
1.500
2.000
2.500
3.000
3.500
t [sec]
Speed (ACSL)
Speed (VTB)
Speed (Solver)



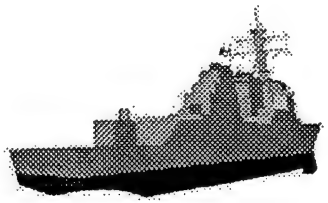
Discussion

- Flat solution space means all equations are solved together - conventional simulation approach
- There is a single solver thread
- This avoids complexity, but may not be the best solution in the long run



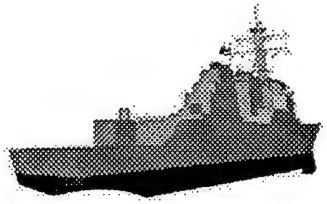
Discussion

- Modular solution space allows multiply threaded solvers
  - Note that single solver thread can be used within a multiply threaded environment since the visualization processes may have multiple threads
- Will be one focus of year two effort



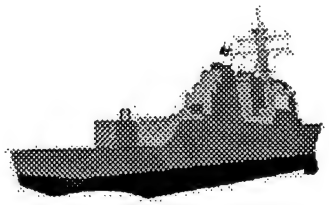
Numerical Issues

- Explicit versus implicit differential equation solvers
  - Implicit methods offer several advantages:
    - Stiff differential systems
    - Differential-algebraic systems
    - Numerical stability



Numerical Issues

- Year two will focus on differential and differential-algebraic solvers using implicit methods
  - Trapezoidal rule
  - Gear's method
  - Petzold's DAE solver (DASSL)

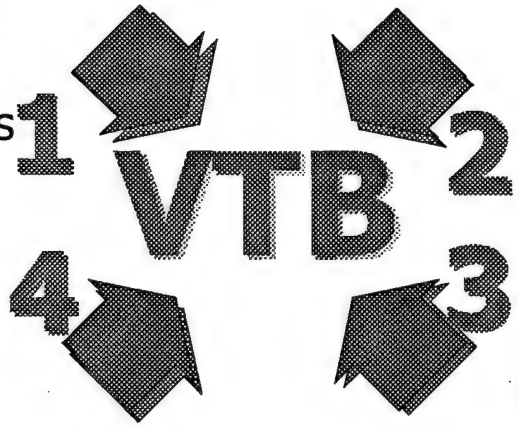


Where We Are Going

- Full Object-Oriented Solvers
  - Modular model and solution spaces
  - Encapsulation
  - Polymorphism
- Multiple Threads
  - Massively parallel solutions
  - Real time system simulations

ASSESSMENT

Reduced Cost by Reuse of Existing Models



Top-Down Mission-oriented Approach

Advanced Visualization Techniques

Network-Oriented Platform independent Computing

1. MODEL REUSE

- Architecture has been defined.
- Comprehensive implementation of one tool (ACSL) with a probe implementation of SPICE.
- Provisions exist for tool-dependent features.
- SIL supports inter-module communication.

3. NETWORK COMPUTING

- VTB implementation imposes no barriers.
- Parallel system is used as lead platform in development.
- High percentage (close to 90%) of VTB code is portable.
- All platform-dependent code is native.

4. VISUALIZATION

- Look and feel is implementation independent.
- All visualization features are portable to both supported OS (Unix, NT).
- An API has been defined for visualization routines.

FUTURE DIRECTIONS

- Additional supported languages
- Formalization of SIL
- Concurrency support
- User input on interface design
- Direct visual programming
- Begin building component library

SOFTWARE ENGINEERING

A large part of VTB is software. Management of the software effort is of considerable importance:

- One of the PI's has become certified to teach the SEI PSP course. PSP training of VTB personnel will begin this summer.
- We will be doing a CMM analysis on our software processes at the beginning of the second year.

UNIVERSITY OF
SOUTH CAROLINA

Modeling of Advanced PEBB Switching Devices

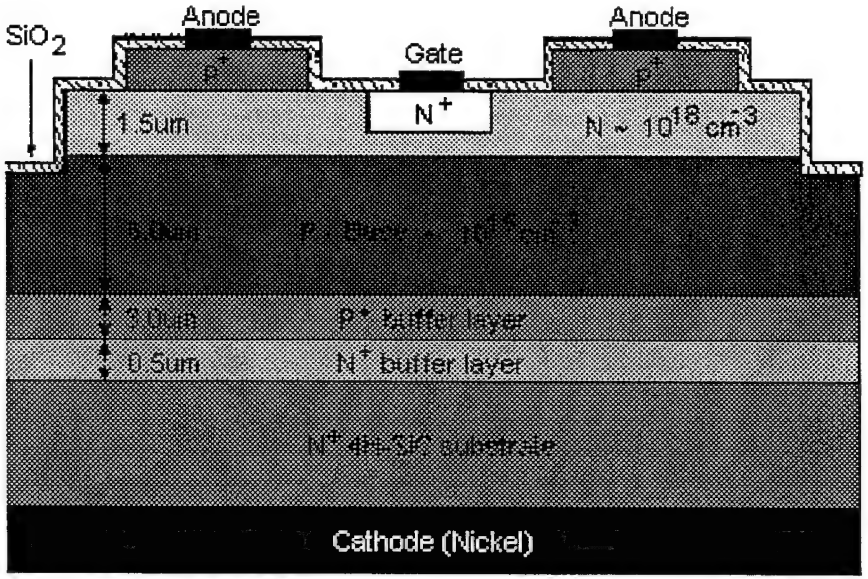
Characterization and modeling of an asymmetrical complementary 4H-SiC GTO

Why model SiC devices now?

- Model of SiC GTO does not exist.
- Modeling provides a tool to develop the next generation of SiC GTOs.
- Test benefits of SiC devices in system simulations.
- Model of small device scales to larger devices.

Device Structure

- 600V Blocking
- Five layers
- Heavily doped n-type 4H-SiC substrate
- Operation at 350°C

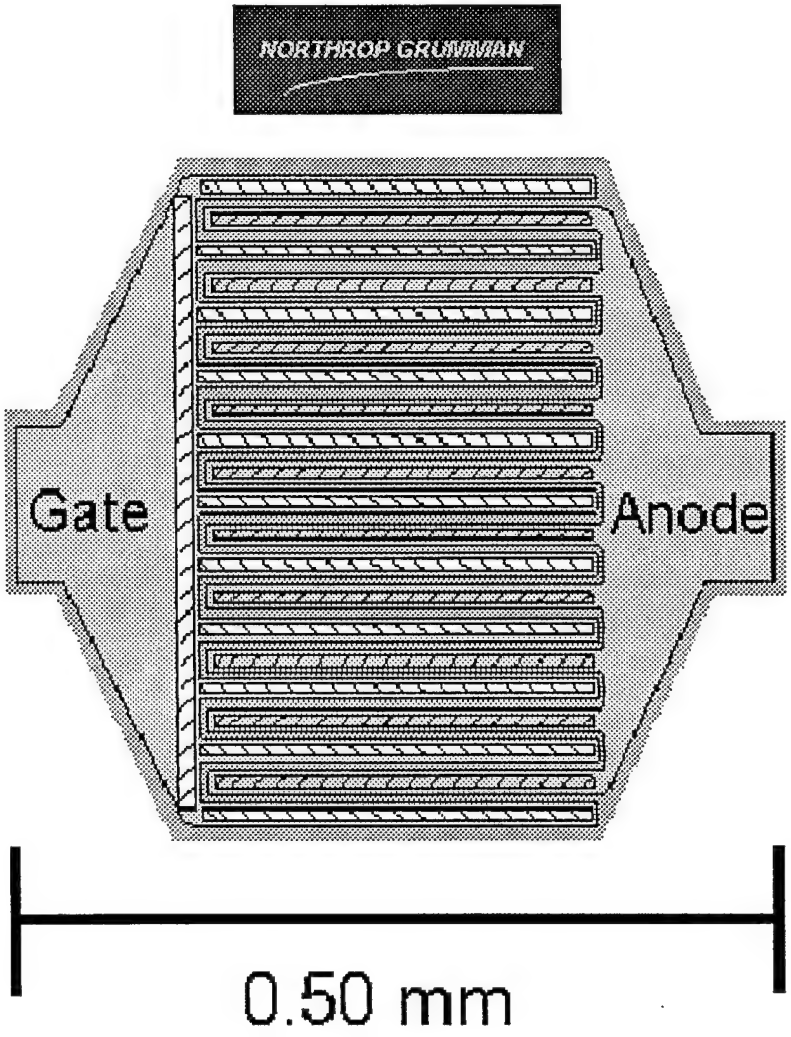


NORTHROP GRUMMAN

UNIVERSITY OF SOUTH CAROLINA

Device Structure

- Twenty finger Gate-Anode Interdigitization
- Anode area = $750 \times 10^{-6} cm^2$
- Capable of 700 A/cm^2
- Implanted guard rings
- No optimization
- Less than 1μs turn-off



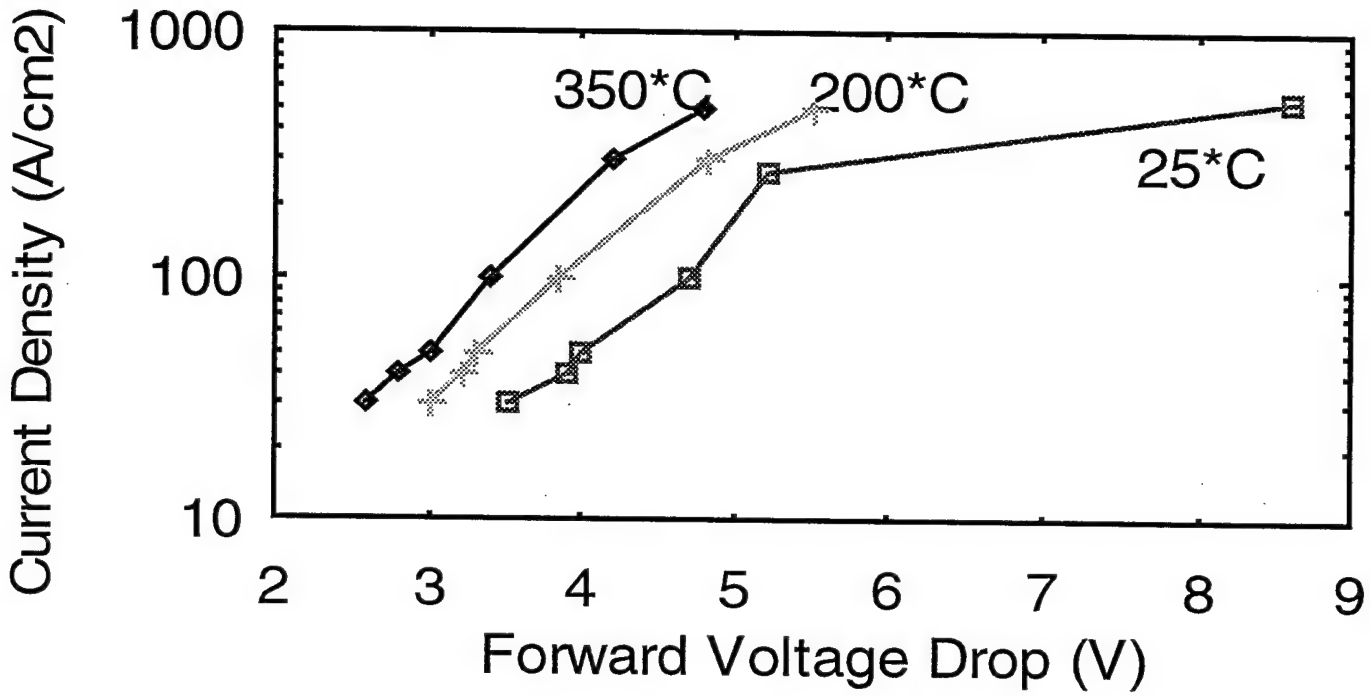
UNIVERSITY OF
SOUTH CAROLINA

Measured characteristics

t_{Doff} Turn-Off Delay	(ns)	52
t_{Don} Turn-On Delay	(ns)	640
t_R Rise Time	(ns)	300
t_F Fall Time	(ns)	360
Max. dv/dt	(V/μs)*	650

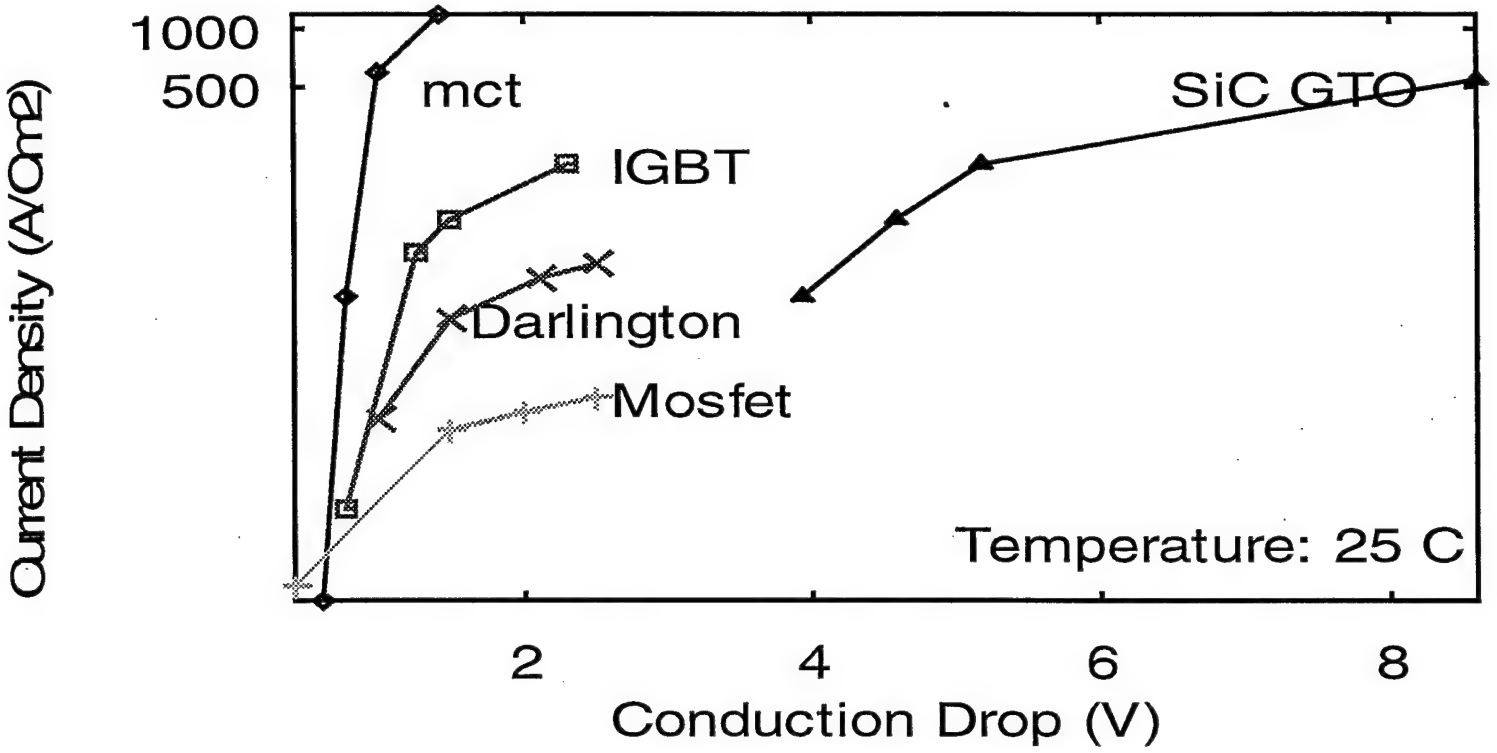
* Measured according to EIA Standard RS-397-1 for JEDEC Registration

Measured characteristics



Forward drop decreases at higher temperatures

Comparison to Silicon devices



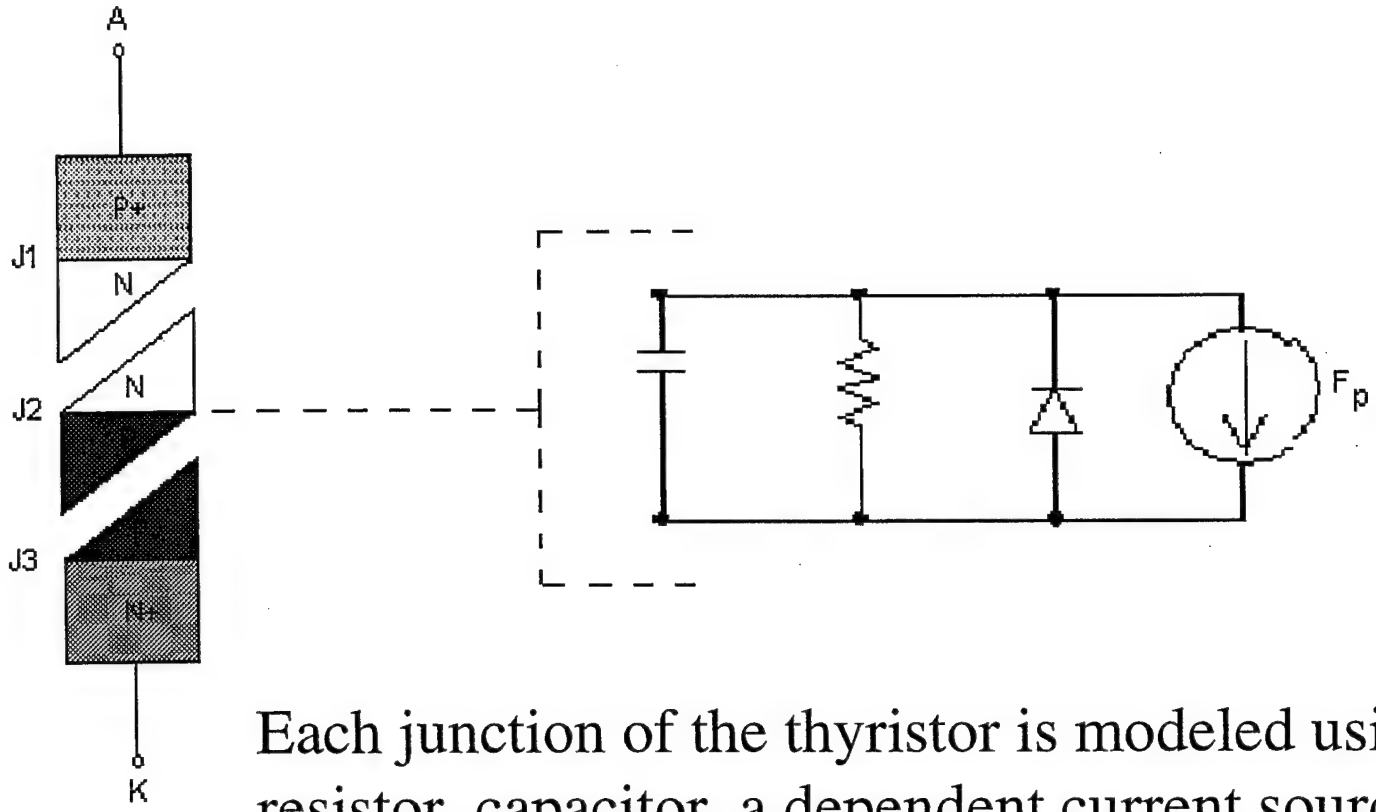
Cannot compare at high temperatures where only SiC devices function.

UNIVERSITY OF
SOUTH CAROLINA

Present Model Capabilities

- Behavioral model
- Turn-on & Turn-off characteristics
- Scalable

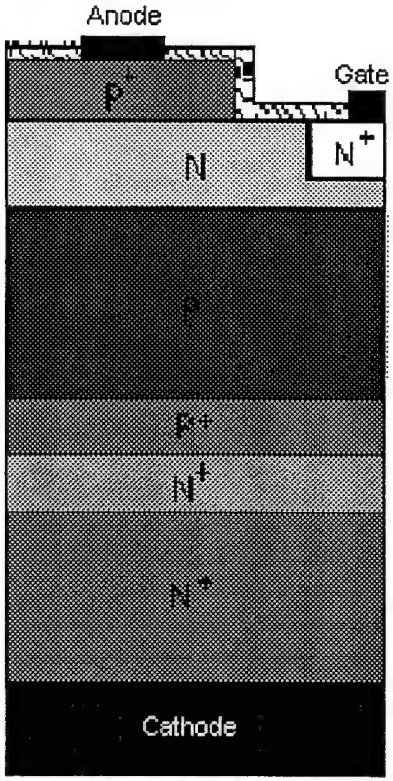
Junction Model

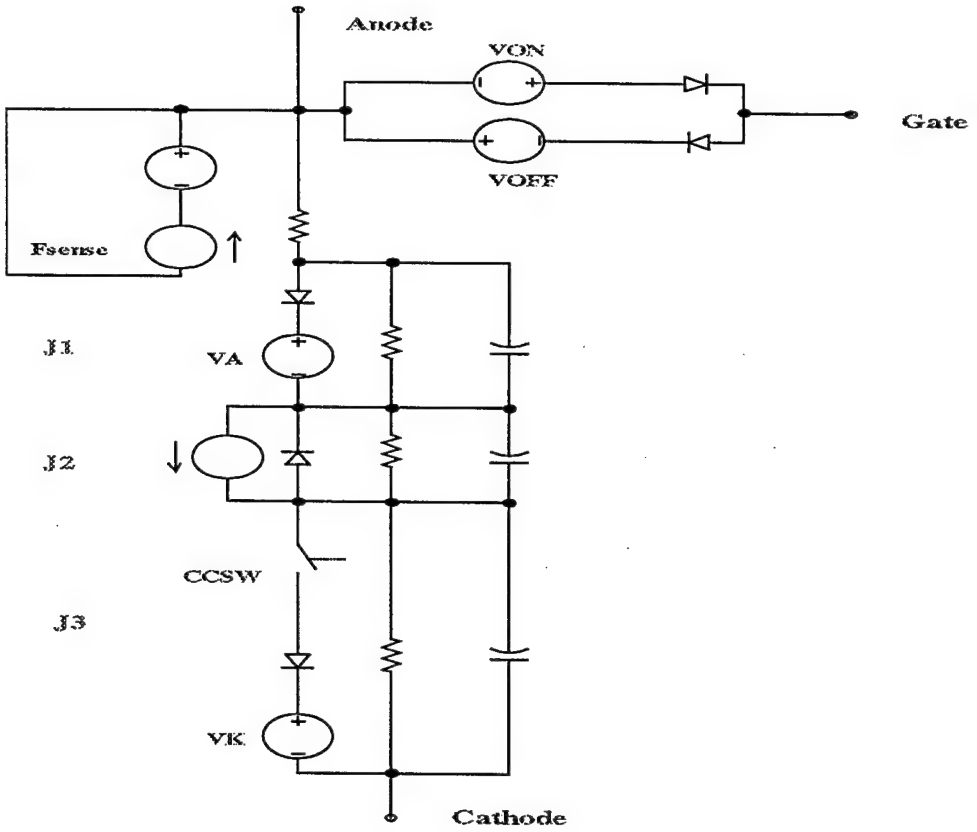


Each junction of the thyristor is modeled using a resistor, capacitor, a dependent current source and a diode. (Modified Ebers-Moll Model)

Model based on - S.Yuvarajan, D.Quek, J.Weimer, "Switching Characteristics and Pspice Model of an MCT," IEEE Trans. pp 1208 - 1215, 1993.

Full Model of GTO





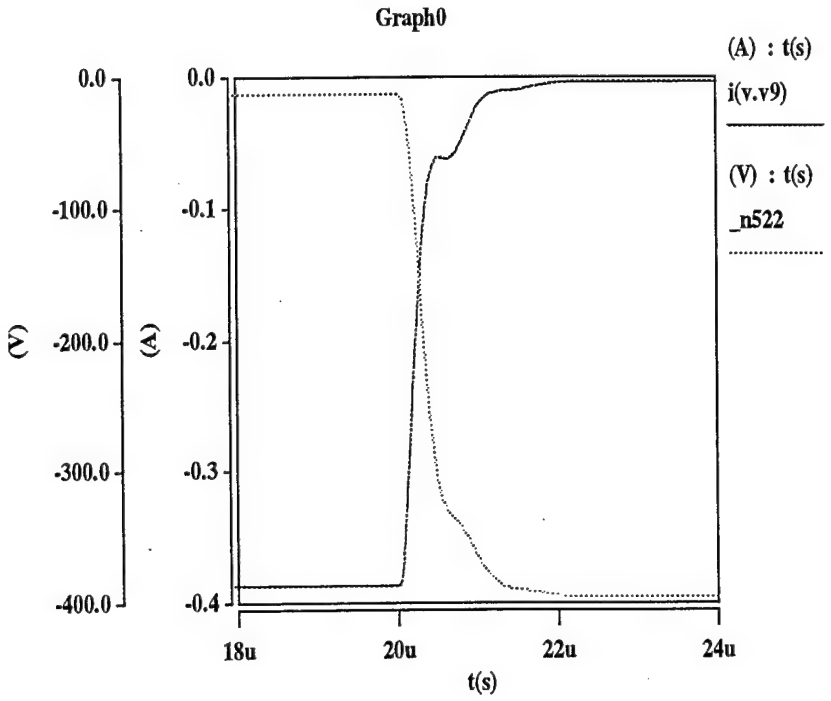
Gate action is modeled using a current controlled switch.

UNIVERSITY OF SOUTH CAROLINA

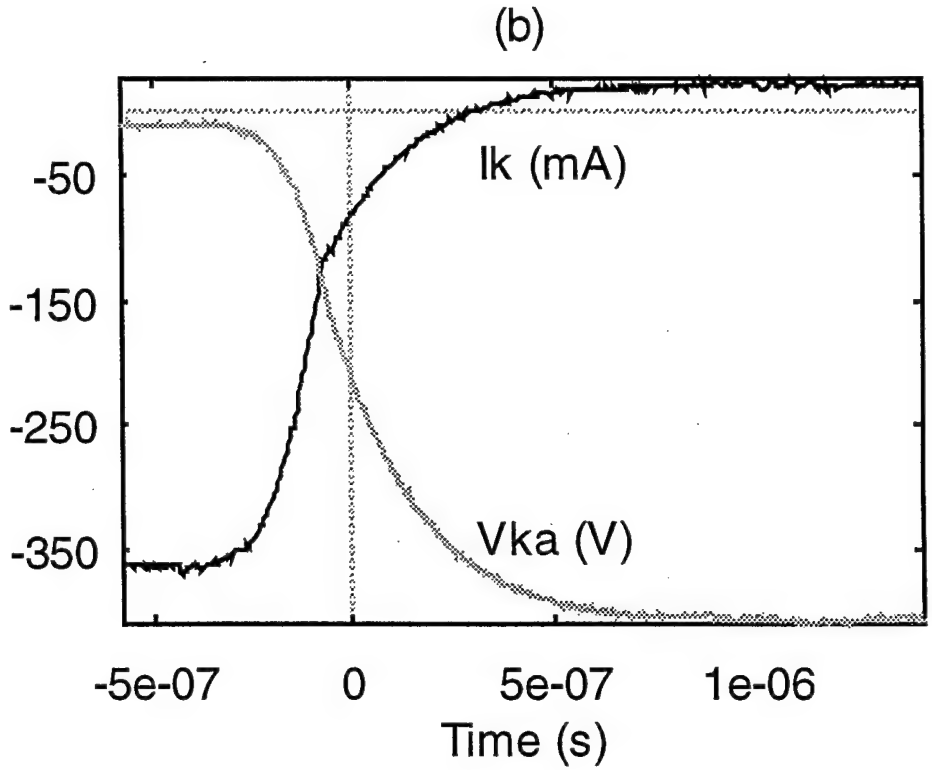
SABER$_{TM}$ Model Parameters

Area	A	1.5×10^{-3}	cm^2
p-n Junction Capacitance	C_j	144.83	pF
Transition Time	t_t	93.54	ps
p-n Junction Potential	V_j	0.77	V
Forward Series Contact Resistance	R_s	22.6	Ω
Reverse Series Contact Resistance	R_z	900	kΩ
Bandgap Energy	E_g	3.86	eV
Electron Mobility	μ_n	500	$cm^2/V \cdot s$
Hole Mobility	μ_p	115	$cm^2/V \cdot s$

Model Performance



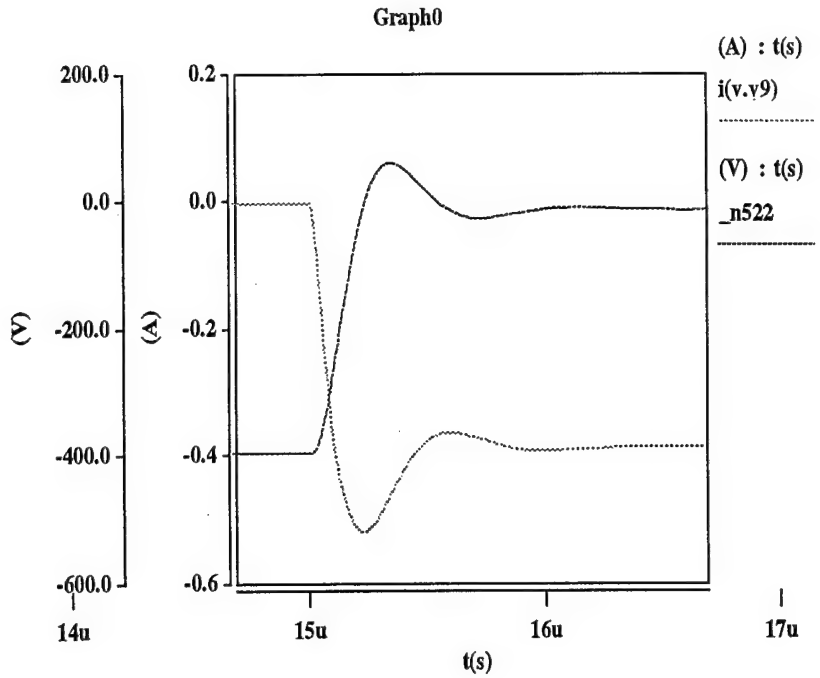
Simulated Turn-off characteristics



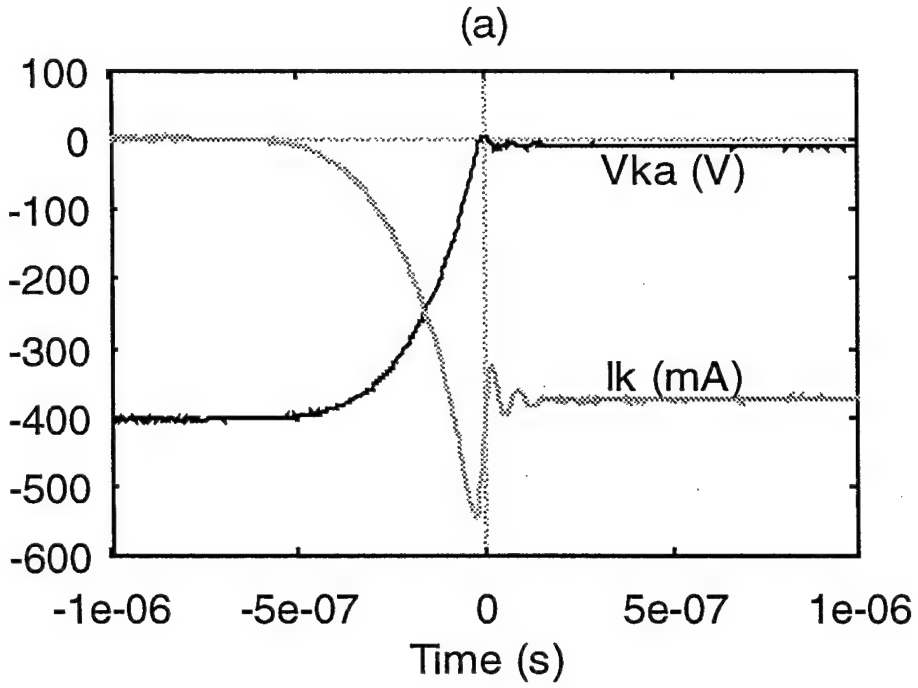
Measured Turn-off characteristics

UNIVERSITY OF SOUTH CAROLINA

Model Performance



Simulated Turn-on characteristics



Measured Turn-on characteristics

UNIVERSITY OF
SOUTH CAROLINA

Future Enhancements

- Include temperature dependencies
- Include package parasitics
- Refine parameters
- Stress analysis from thermal cycling
- Create physics-based model

Conclusions

- Measurements were made to extract parameters for a SiC GTO circuit model
- A behavioral model was developed on $SABER_{TM}$ to study the switching characteristics
- The device model can be scaled up to a larger one for evaluating it's usefulness in a PEBB power Module

PEBB Applications:

Extraction of Package Parasitics
and
Advanced Modeling of PCBs

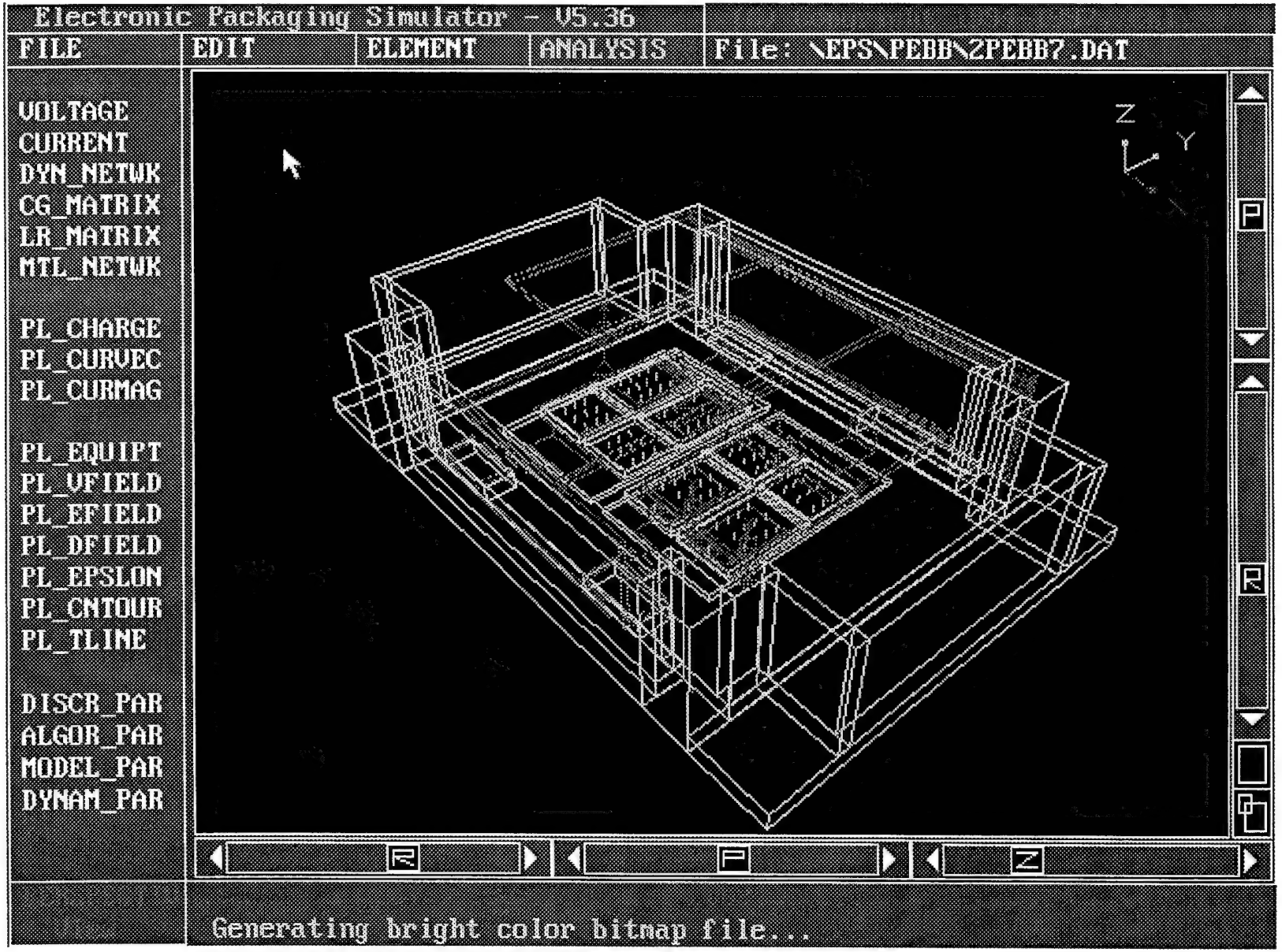
Goals for VTB technologies:

- tools for electrical parameter extraction
  - integrated with other parts of VTB
  - fast simulation capability
  - “interactive”

Capabilities of existing extraction tools:

- stand-alone parameter extraction tool:
  - RLC matrices or N-port network transfer functions for:
    - arbitrary lossy/lossless 3-D objects
    - lossy/lossless PCB substrates, traces, discontinuities
    - complex packages and Si devices

PEBB modules:



Quasi-Statics:

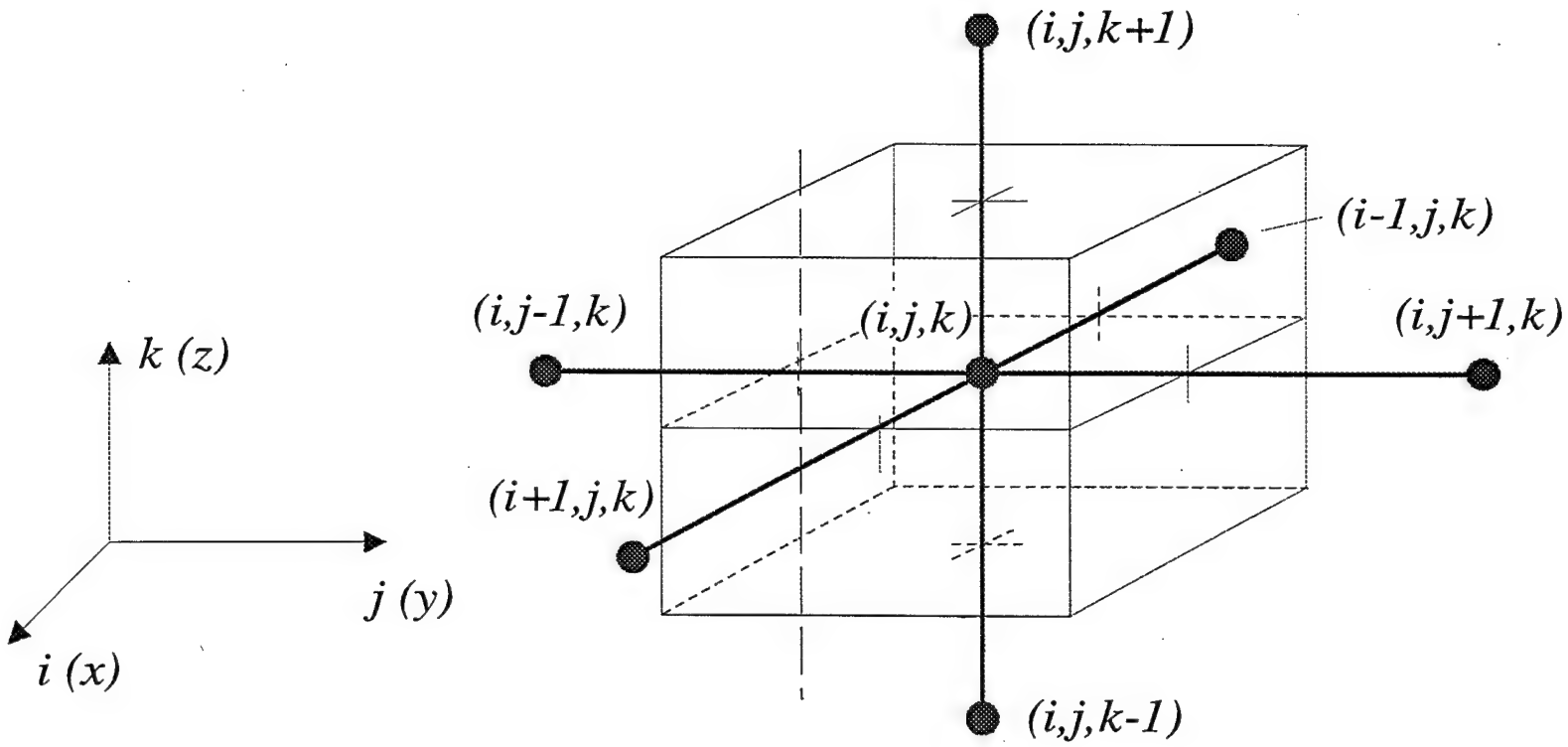
- solution to Laplace eq'ns for voltage

$$\nabla \cdot \left(\begin{Bmatrix} \varepsilon_r(x, y, z) \\ \sigma(x, y, z) \end{Bmatrix} \nabla V \right) = 0$$

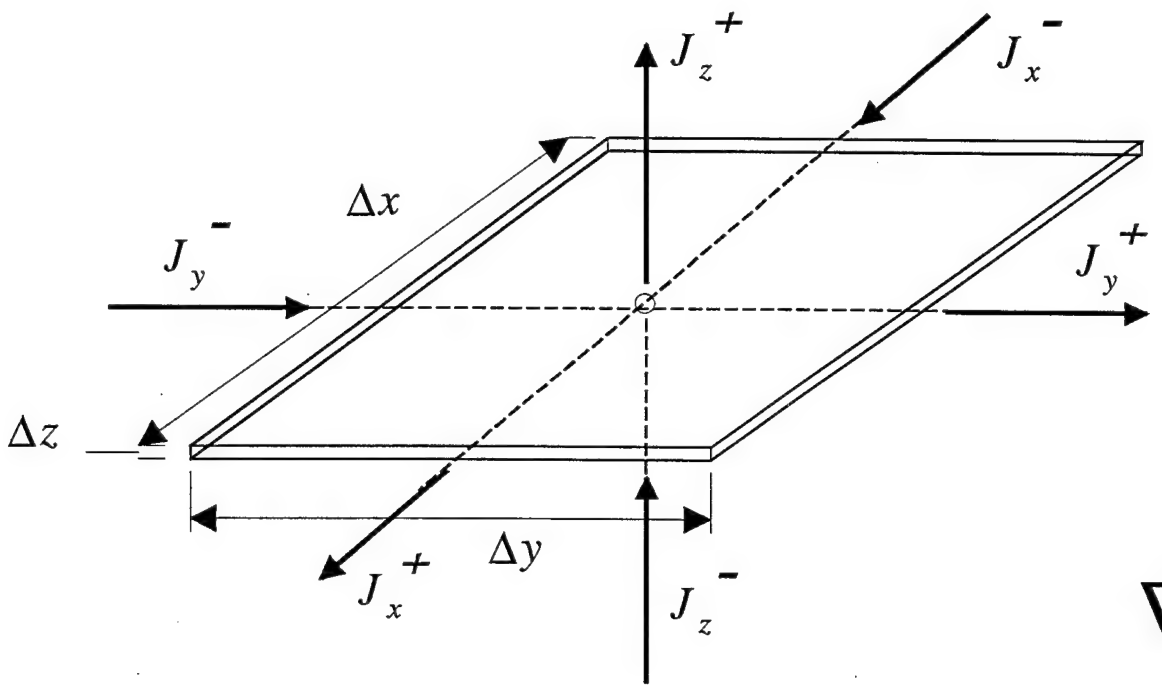
- Gauss law for capacitance & conductance

$$\begin{pmatrix} C \\ G \end{pmatrix} = \frac{\oint_S \begin{pmatrix} \varepsilon(x, y, z) \\ \sigma(x, y, z) \end{pmatrix} \vec{E} \cdot \hat{n} ds}{V_{diff}}$$

Finite Difference Method: Laplace eq'n



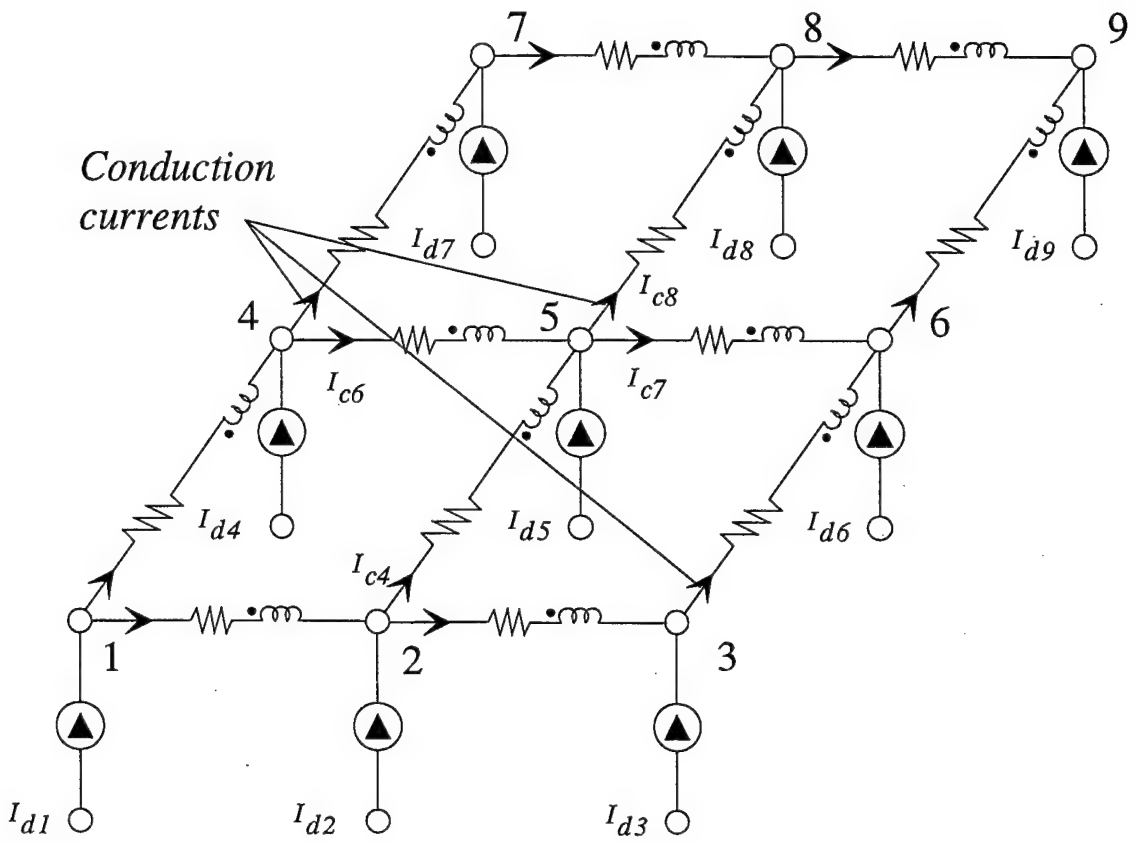
Current Simulation Method:



$$\nabla \cdot \vec{J} = -j\omega\rho$$

$$\int_V \nabla \cdot \vec{J}_T dv = \oint_S \vec{J}_T \cdot d\vec{S} = \sum_{k=1}^{K} I_k = 0$$

Continuity equation to network form:



$$\Delta y \Delta z \left(J_{cx}^{+} - J_{cx}^{-} \right) + \Delta x \Delta z \left(J_{cy}^{+} - J_{cy}^{-} \right) + j\omega \Delta x \Delta y \left(D_{z}^{+} - D_{z}^{-} \right) = 0$$

Inductance and resistance:

- partial inductance for *L*
- skin resistance for *R*
- effect of non-uniform current on *L* and *R*

$$L_{eff} = \frac{2W_m}{I_{port}^2} = \frac{2\left[\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} I_i L_{ij} I_j\right]}{I_{port}^2}$$

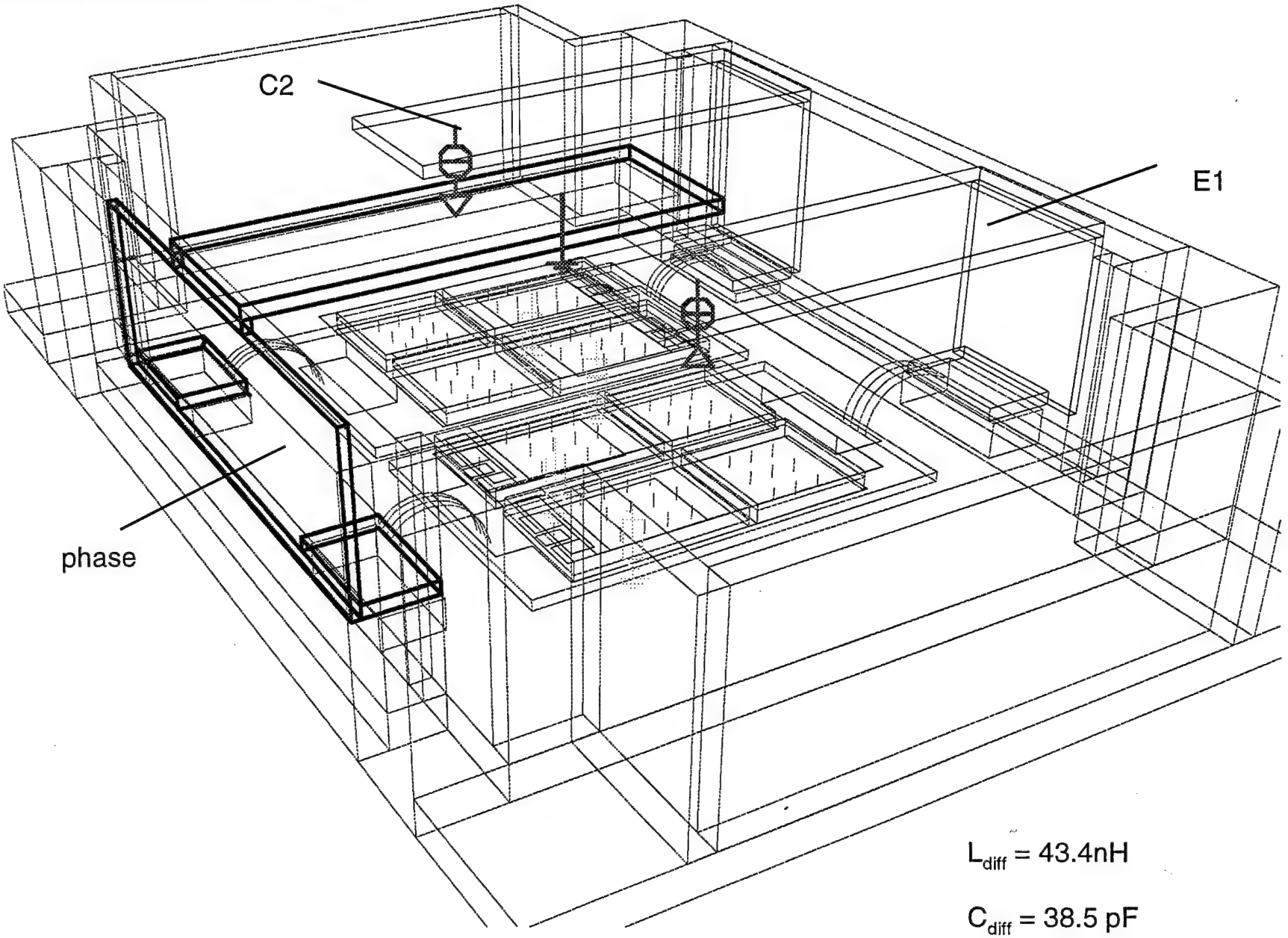
$$R_{eff} = \frac{W_{diss}}{I_{port}^2} = \frac{\left[\sum_{i=1}^{N} R_{ii} I_i^2\right]}{I_{port}^2}$$

Examples of past use

- Post-production simulation of PEBB1:
  - to validate experimental set-up and data
- Pre-production simulation PEBB1A:
  - to "virtually" test originally proposed design
  - to evaluate new design alternatives
  - to check effects of thermal enhancements on electrical performance

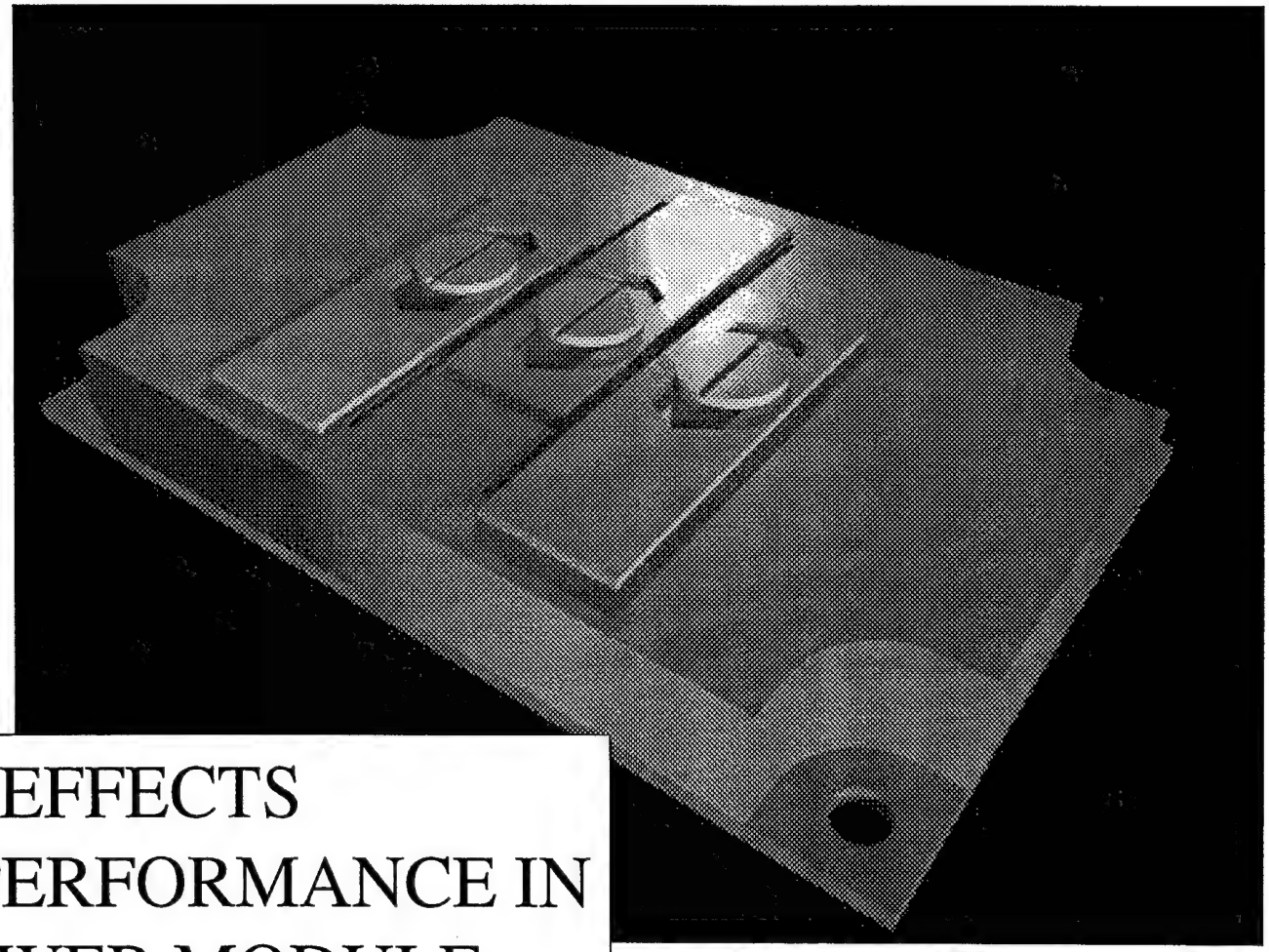
Movie time, let's get the popcorn...

PEBB2a design with Tabs:



L_{diff} = 43.4nH

C_{diff} = 38.5 pF



PARASITC EFFECTS ON ARCP PERFORMANCE IN A PEBB POWER MODULE

Why Model the PEBB Power Module?

- To Study the effects of power module package parasitics on ARCP performance
- To study device losses on ARCP performance
- To evaluate the performance of advanced switching devices
- Allows feasibility studies for new PEBB Applications

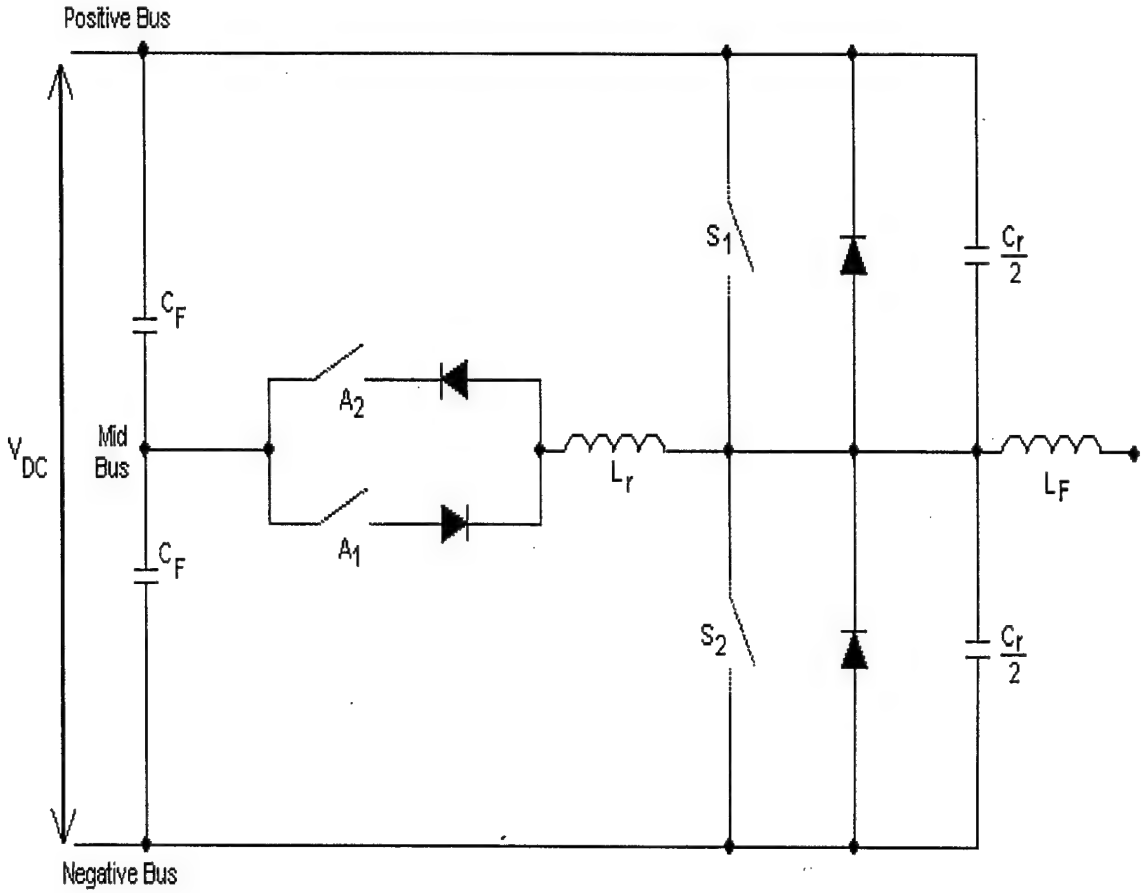
UNIVERSITY OF SOUTH CAROLINA

What is ARCP?

- Auxiliary Resonant Commutated Pole
- Soft switching circuit topology
- Offers significant reduction in switching losses
- Higher switching frequency operation
- Zero-voltage turn-on of main switches
- Zero-current turn-off of auxiliary switches

Ideal ARCP - Phase Leg

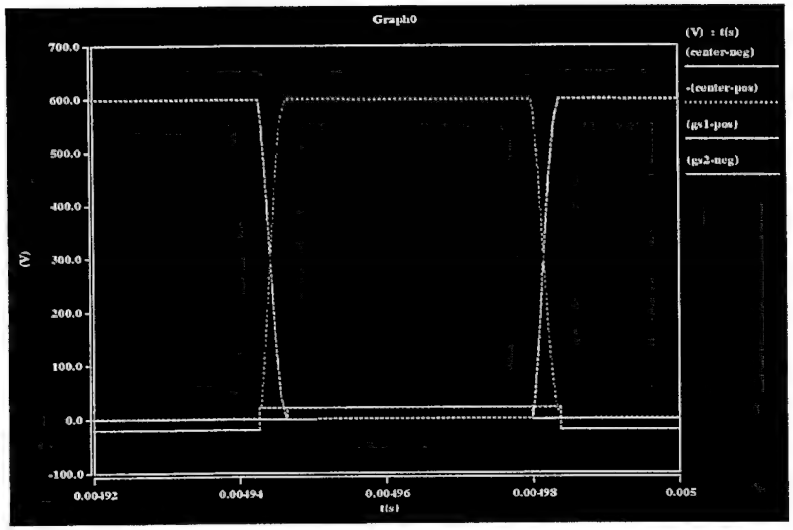
- DC Bus at 600V
- Operating as a 15KHz Square wave inverter
- Resonant Inductor Lr = 3µH
- Resonant Capacitor Cr = 25µF
- Ideal switches and diodes

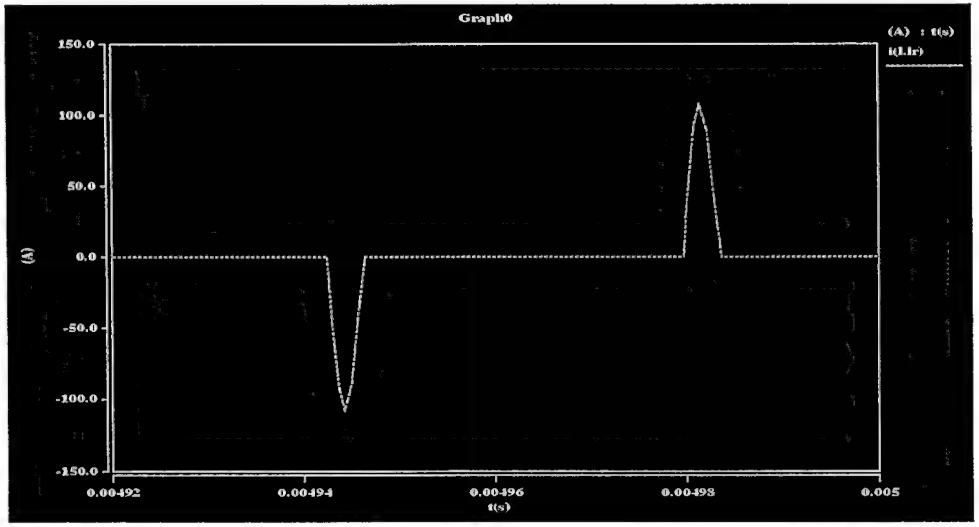


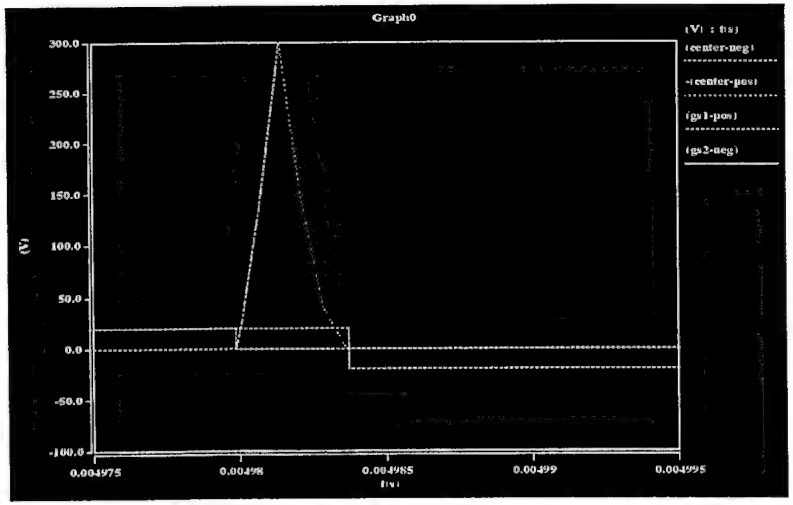
Refrence: R.W.De Doncker and J.P.Lyons, "The Auxiliary Resonant Commutated Pole Converter", IEE-IAS, 1990

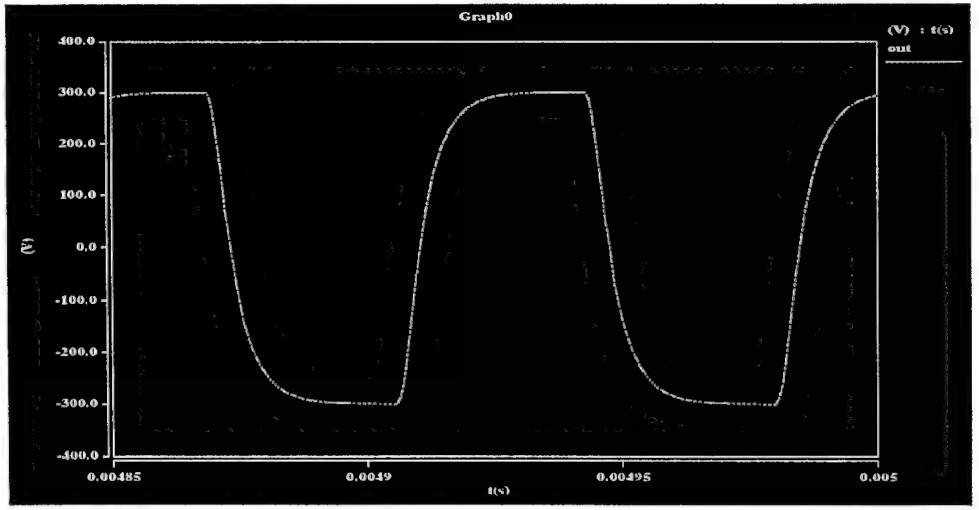
UNIVERSITY OF SOUTH CAROLINA

Ideal ARCP Performance



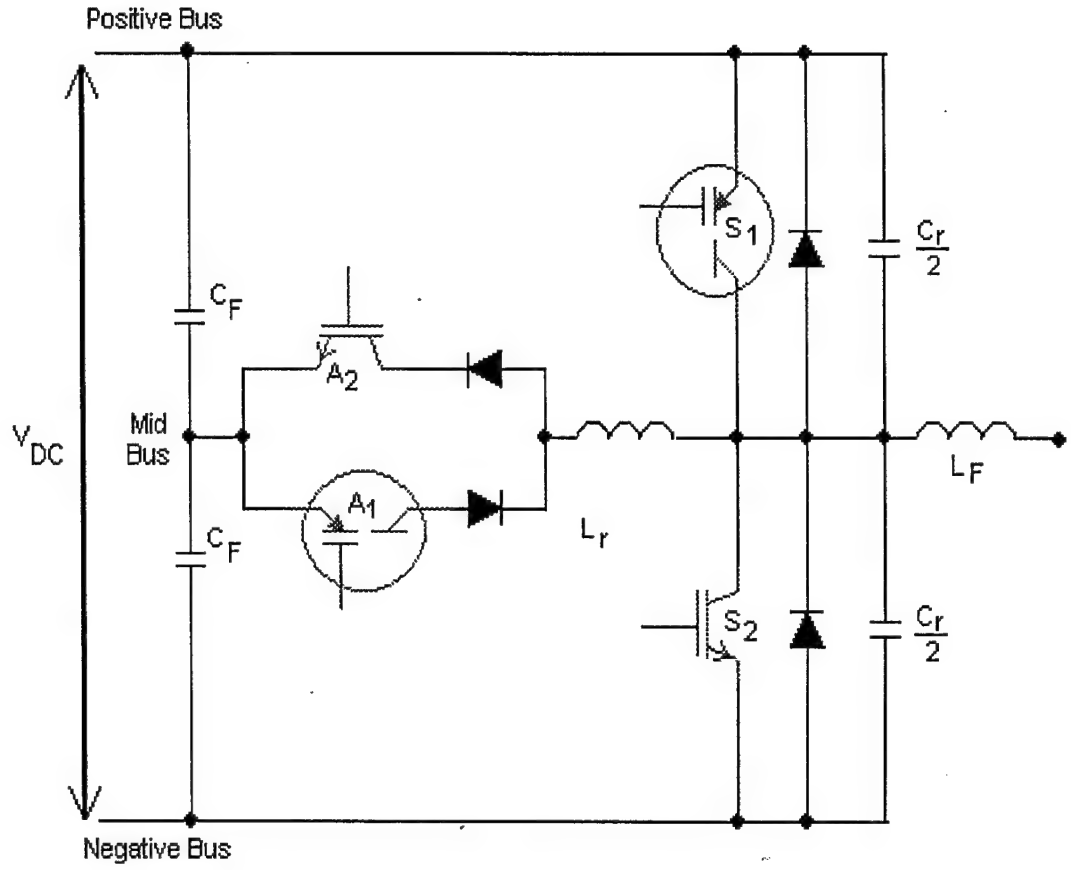






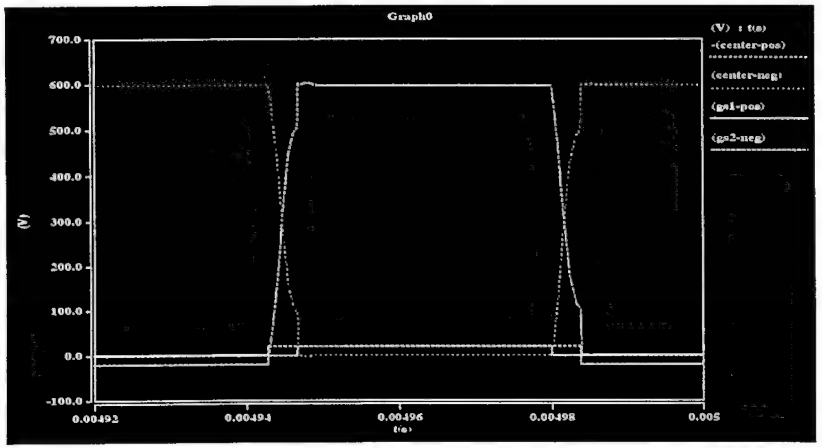
Practical ARCP-Phase Leg

- 600V Harris PMCT
  - I_{AMAX} = 120A
  - Forward drop - 2.2V
  - Turn-off < 1μs
- 600V IR IGBT
  - I_{MAX} = 40A
  - V_F = 3.2V
  - Turn-off < 1μs

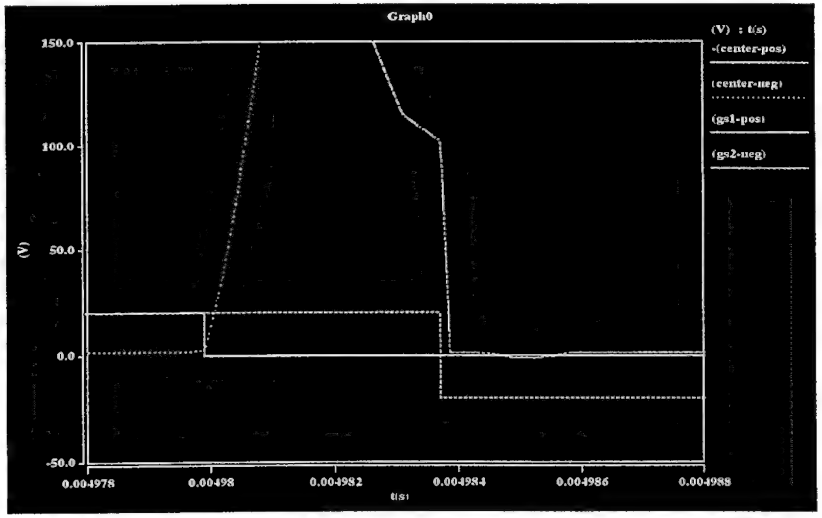


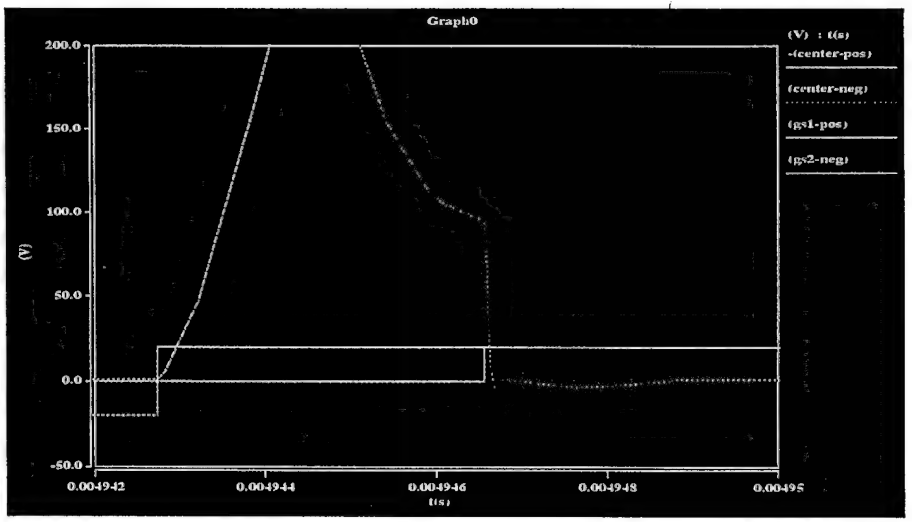
UNIVERSITY OF SOUTH CAROLINA

Practical ARCP Performance



- Main switches no longer show zero voltage switching due to device losses
- Need to compensate for device losses (4.2 I_{load})

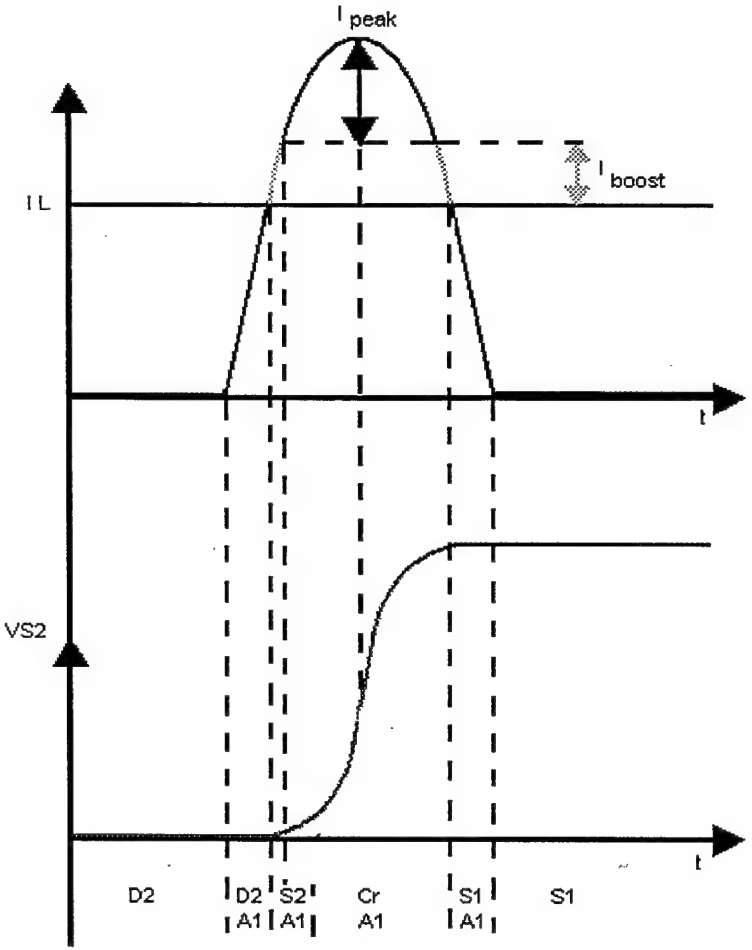




Addition of Boost phase

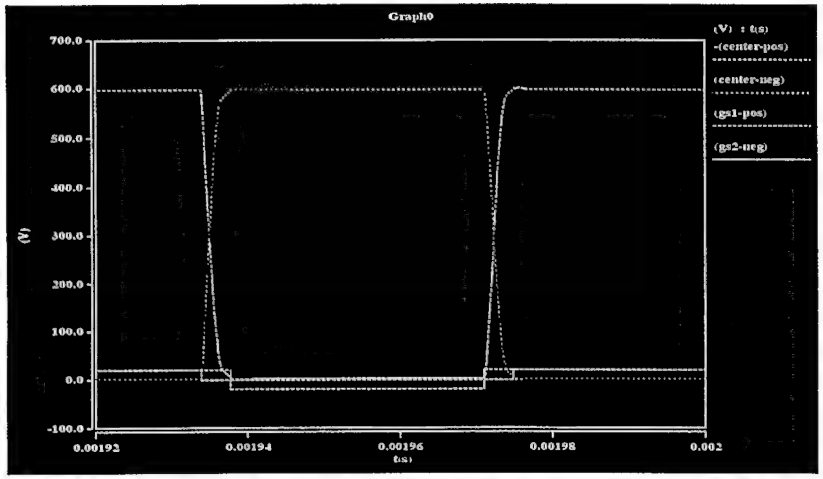
- During the ramp-up phase the the current through the resonant inductor surges past I_{load} to a pre-determined I_{boost} value

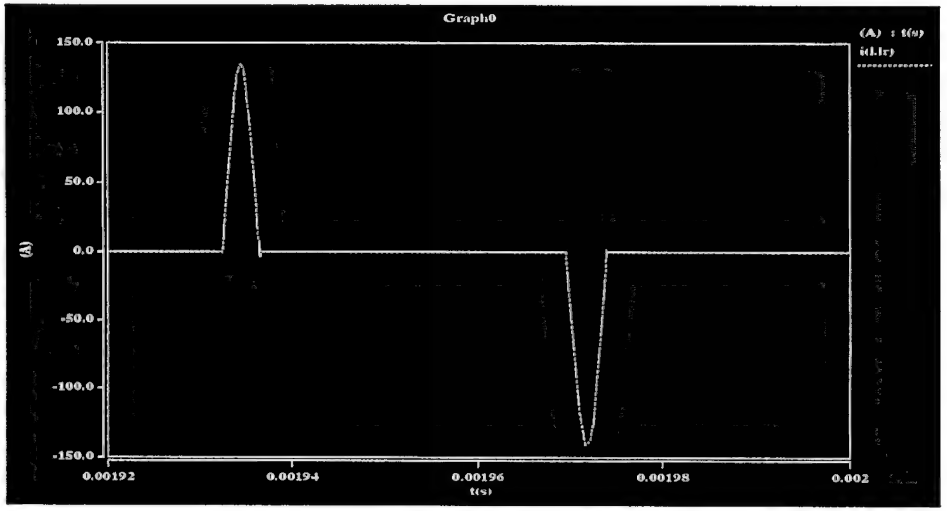
- Auxiliary switches are on for a longer time, thus increasing the time needed to commutate

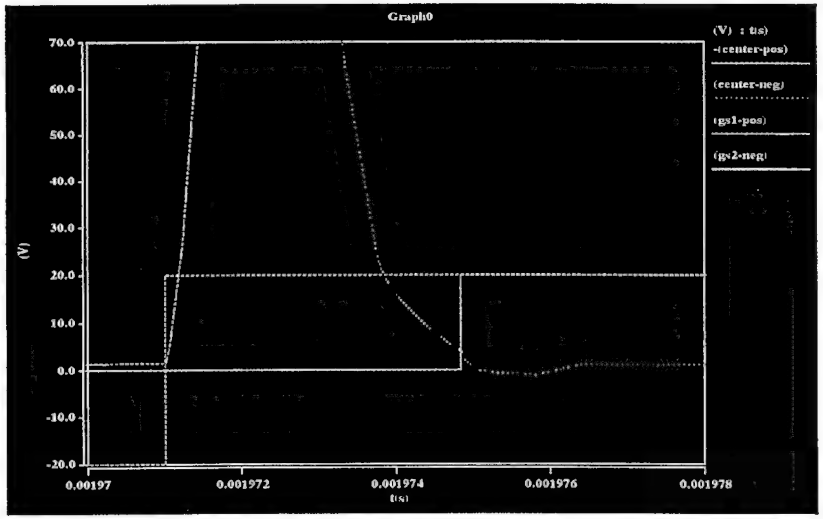


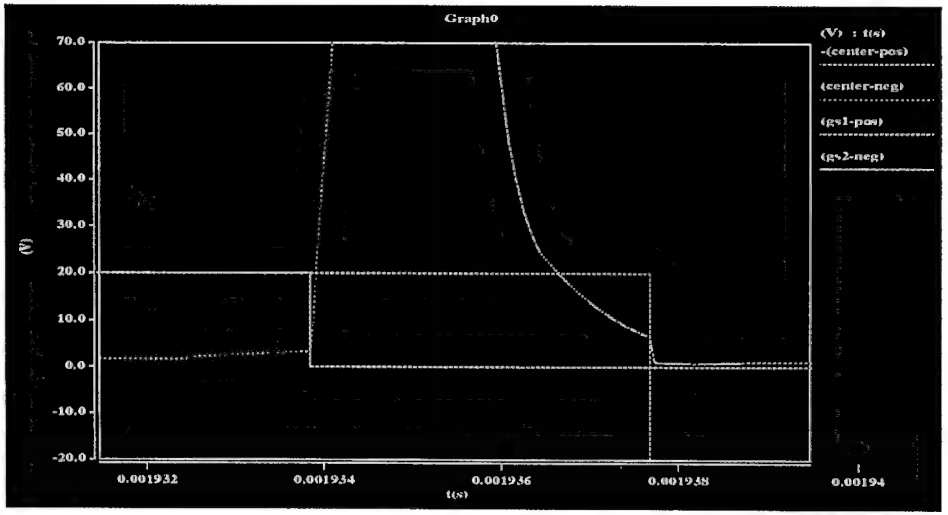
UNIVERSITY OF
SOUTH CAROLINA

Performance with Boost

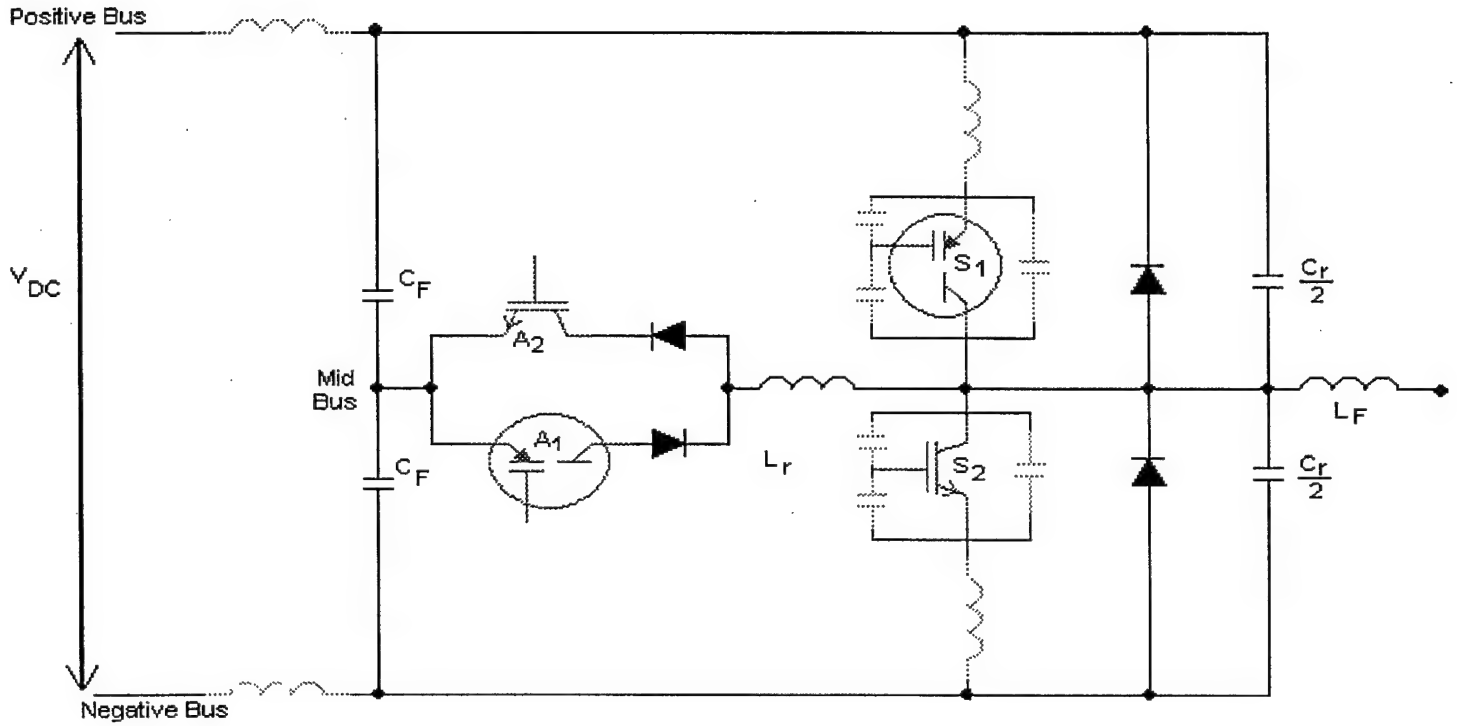








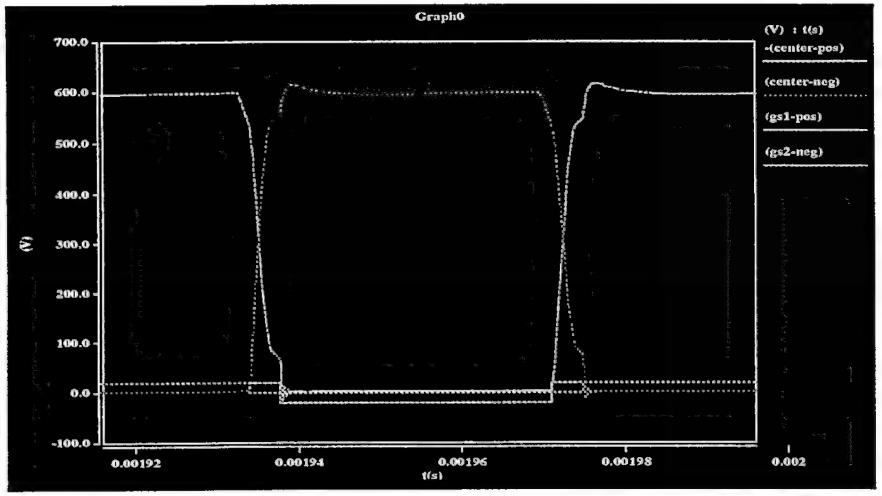
Power Module Parasitics



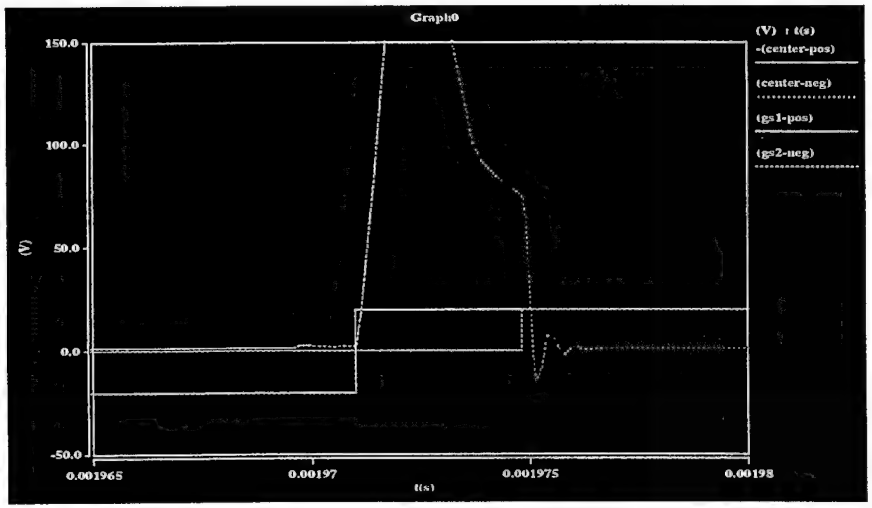
Power Module package parasitics are added to analyze performance

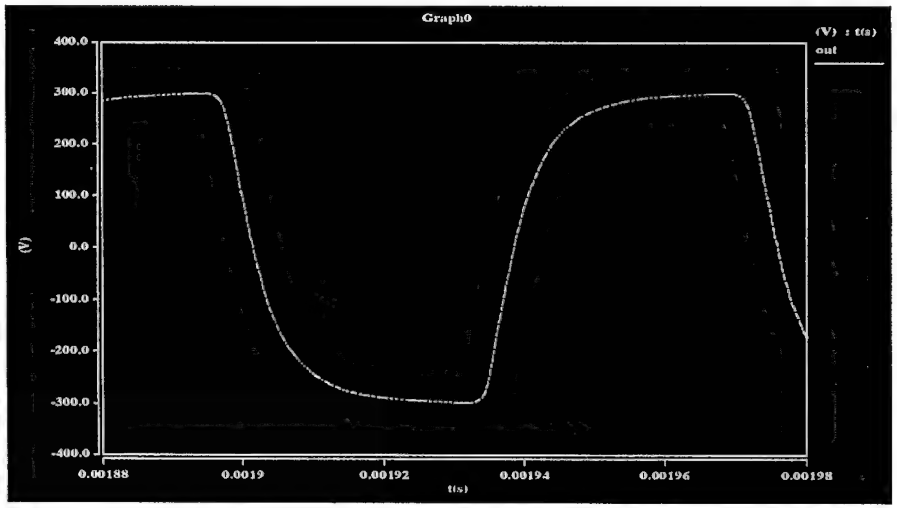
UNIVERSITY OF SOUTH CAROLINA

Performance with Parasitics

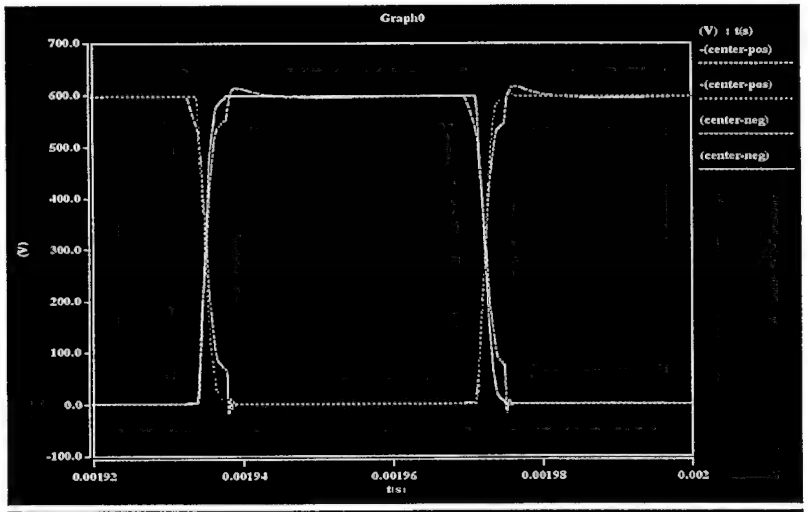


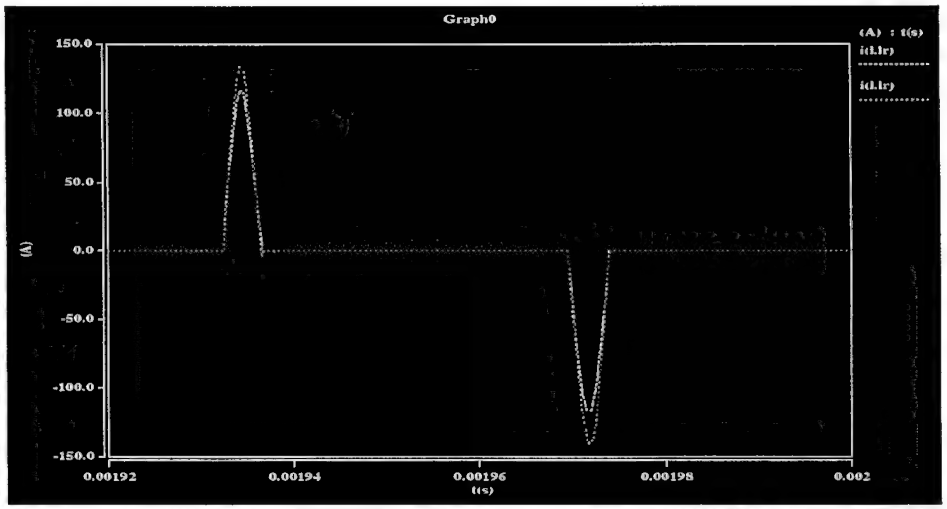
- Non-Zero voltage switching
- Earlier boost does not compensate for package parasitic losses
- Additional boost requirement increases the commutation time and the value of I_{peak}

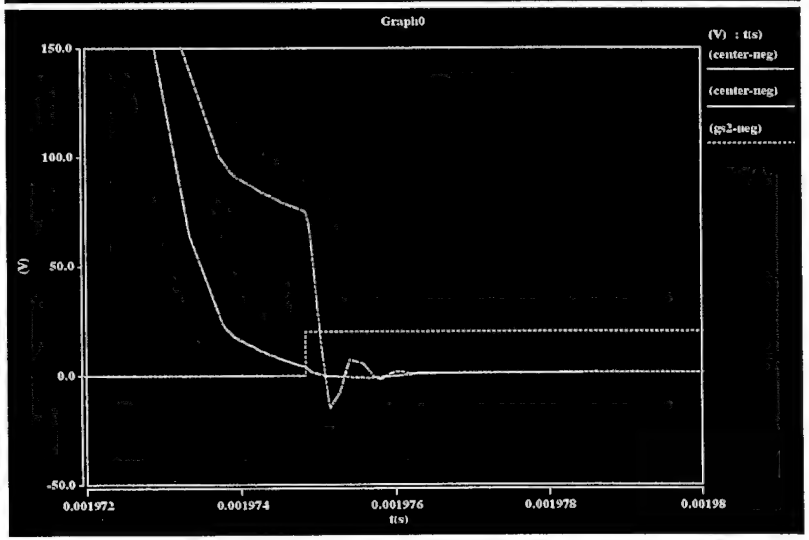


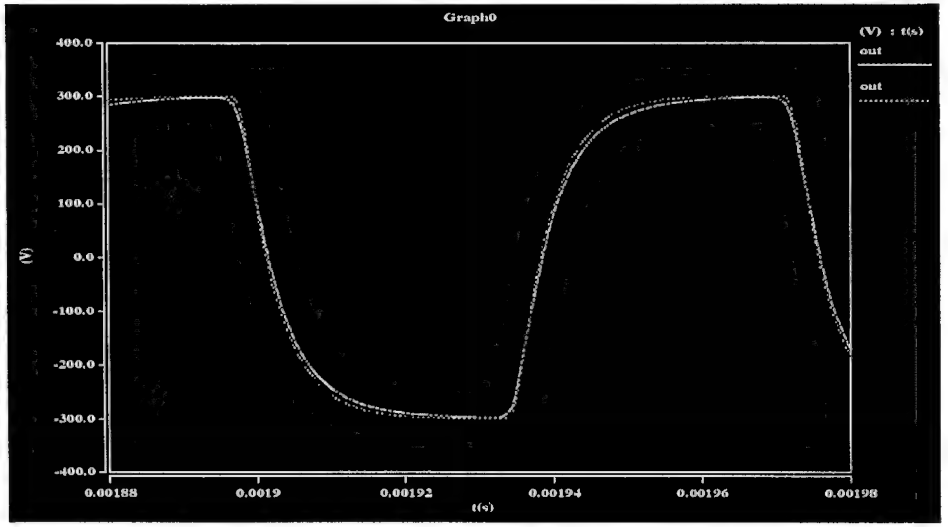


With & Without package parasitics









Conclusion

- The Power Module parasitic impedances have a significant impact on ARCP performance
- Addition of a boost phase can compensate for component and packaging losses
- A boost phase increases the commutation time, thus decreasing the maximum operating frequency
- It also increases the amplitude of the resonant cycle, which could require larger auxiliary switches and resonant inductors.

UNIVERSITY OF
SOUTH CAROLINA

AC MOTOR CONTROL : A SIMULATION STUDY

by
Levent U. Gökdere

IMPLEMENTATION

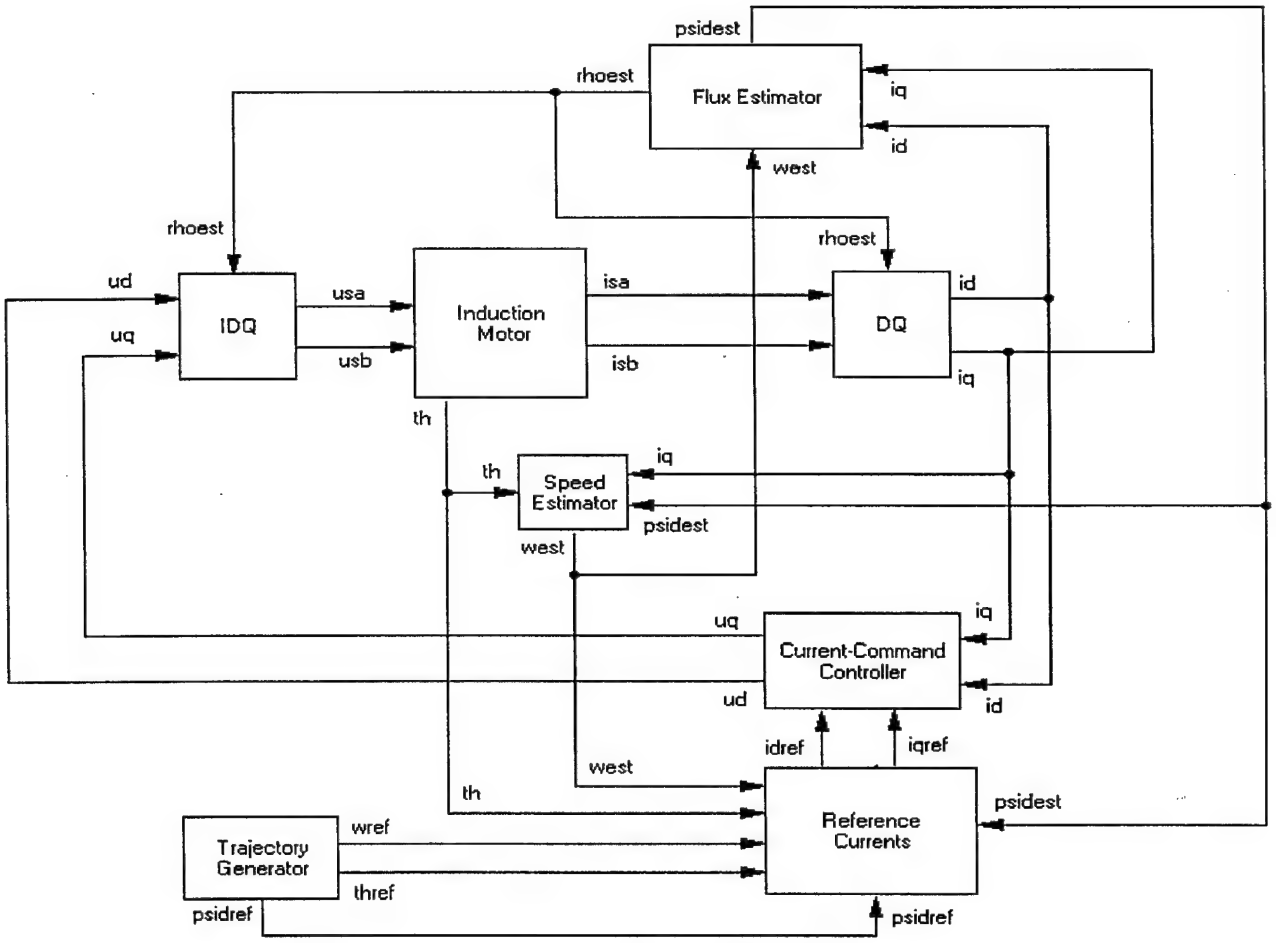
- Field-Oriented Control of Induction Motor Using ACSL Graphic Modeller (ACSL/GM).

- ACSL/GM enables user to design, analyze, and communicate the system in terms of block diagrams.

FEATURES OF FIELD-ORIENTED METHOD

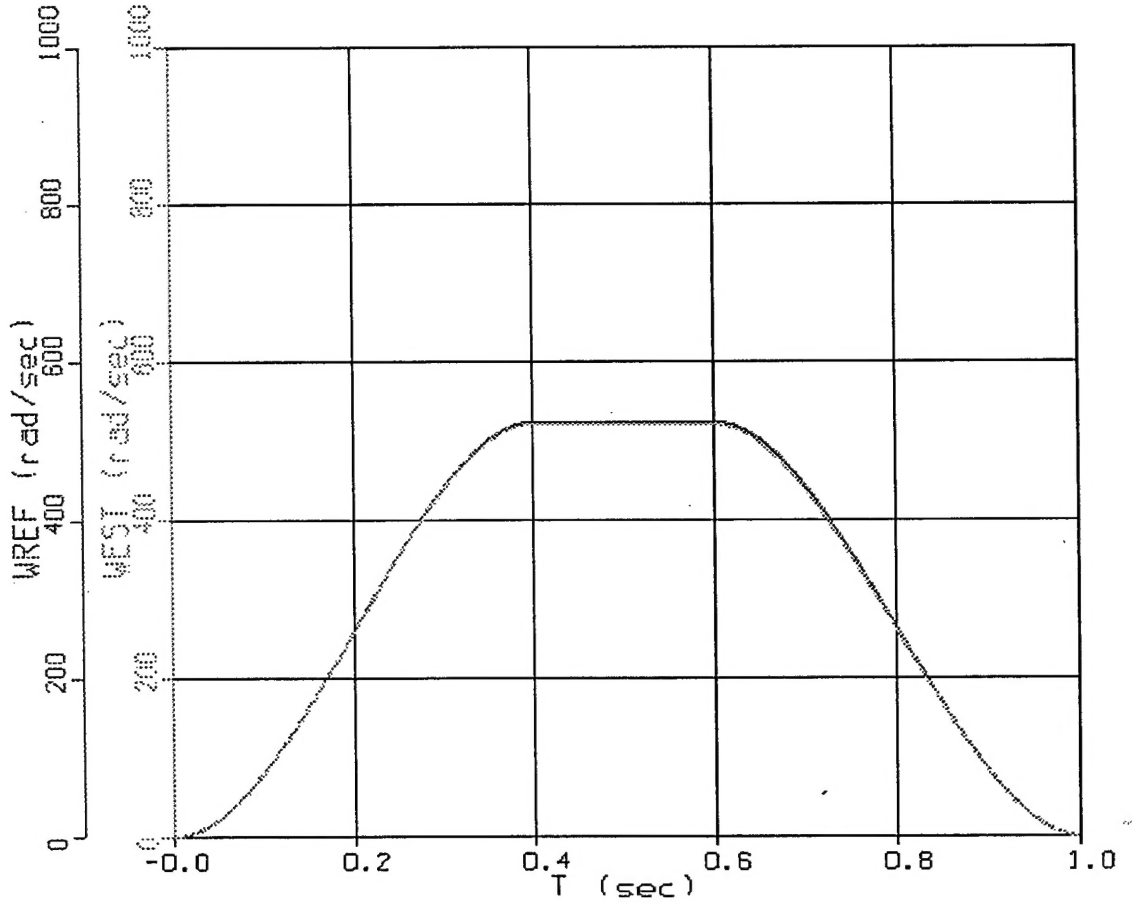
- Implemented in discrete time (sampling frequency = 5 kHz).
- Provides close tracking of time-varying speed/position and flux trajectories.
- Uses estimated (rather than measured) values of speed and flux.
- A drawback: Requires position measurements through an optical encoder.

FIELD-ORIENTED CONTROL OF INDUCTION MOTOR

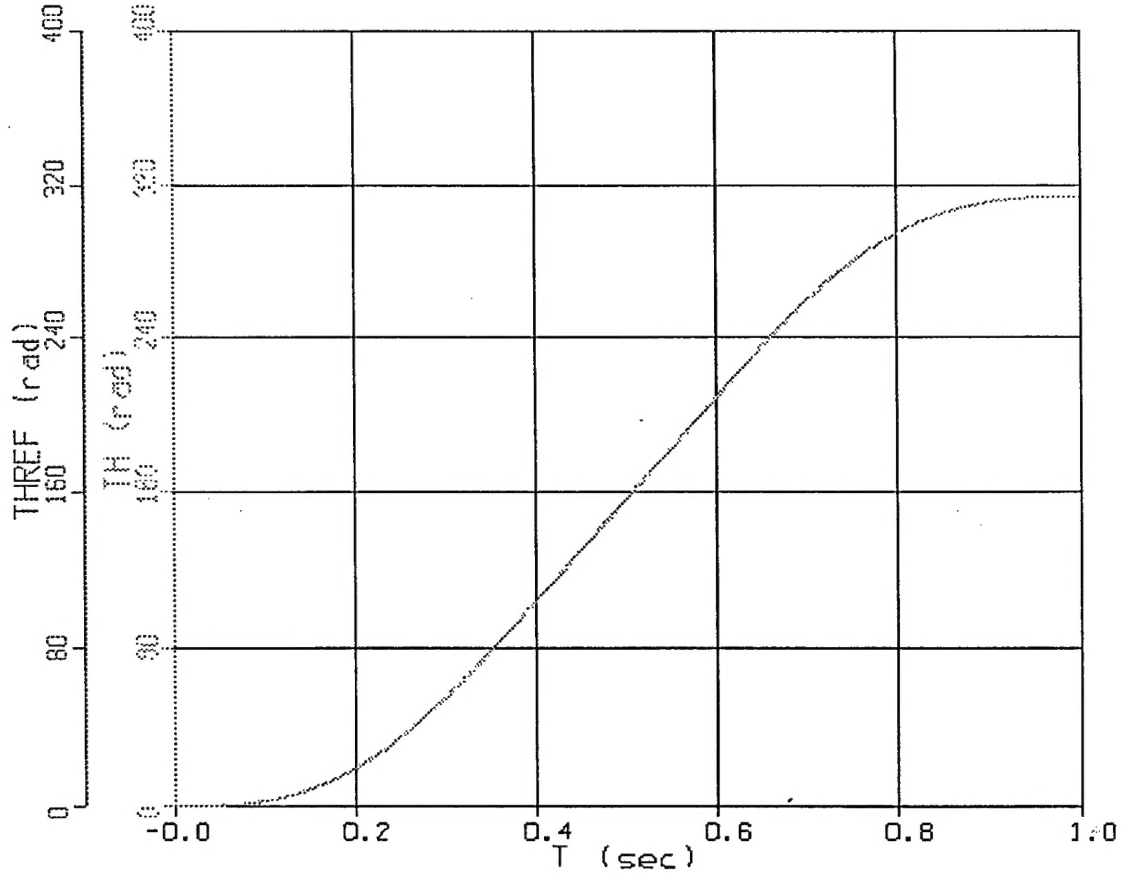


UNIVERSITY OF
SOUTH CAROLINA

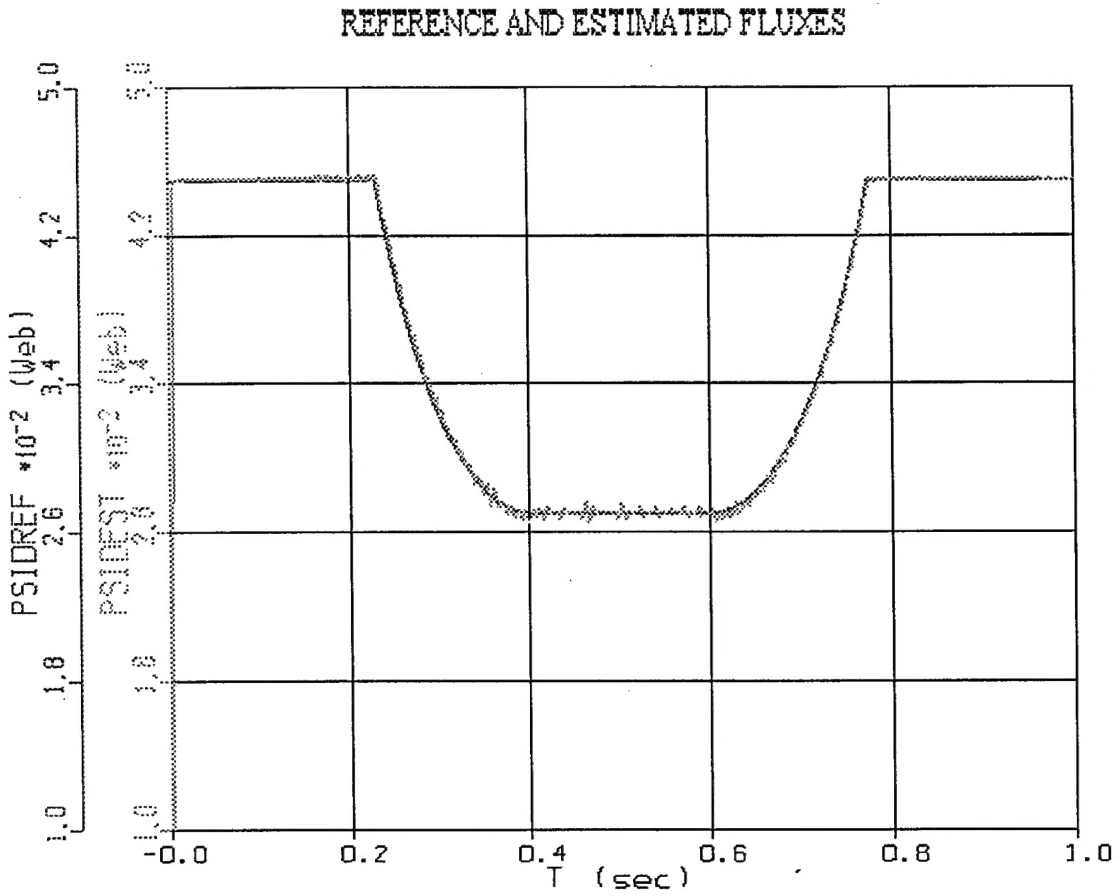
REFERENCE AND ESTIMATED SPEEDS



REFERENCE AND MEASURED POSITIONS



WREF AND THREF ARE RELATED BY WREF = d(THREF)/dt



PSIDREF = psidmax for WREF < wbase
PSIDREF = psidmax (wbase/WREF) for WREF >= wbase
psidmax = 0.045 Webers, wbase = 300 radians/second

UNIVERSITY OF
SOUTH CAROLINA

COMPUTER SIMULATION DEMO IS AVAILABLE